ioProgrammo

ANTEPRIMA SU INDIGO
IL NUOVO FRAMEWORK PER LA CREAZIONE DI SERVIZI CHE COMUNICANO IN REMOTO

Rivista + "Le grandi guide di ioProgrammo" n°5 a € 12,90 in più
VERSIONE PLUS RIVISTA+LIBRO+2CD €9,90
VERSIONE STANDARD RIVISTA+2CD €6,90

PER ESPERTI E PRINCIPIANTI
Poste Italiane S.p.A Spedizione in A.P. • D.L. 353/2003 (conv.in L.27/02/2004 n.46) art.1 comma 2 DCB ROMA Periodicità mensile • GIUGNO 2006 • ANNO X, N.6 (103)

TELEVISIONE OVER IP

Tutti la chiedono. Ecco le cose da conoscere per non farti trovare impreparato

TEORIA Mai più dubbi su Tv On Demand, Broadcasting, larghezza di banda e compressione

STRUMENTI Windows Media SDK, Encoder e Media Services. Impara ad usarli!

PRATICA Crea una trasmissione, aggiungi la pubblicità e gestisci lo streaming

CODICE Un'applicazione completa e personalizzata in Visual Basic

msdn WEBCAST
3 VIDEOGUIDE UFFICIALI PER IMPARARE MICROSOFT VISUAL C# 2005

SQL SERVER 2005 USALO COSÌ!

Rendi tutto velocissimo con le Stored Procedure, automatizza il lavoro con i Jobs, crea DLL in .NET e registrale nel database!

CASI DI STUDIO: SOLUZIONI PER LA TELEFONIA MOBILE

SALVA I DATI SUL CELLULARE

Impara le tecniche per conservare testi e immagini sul telefonino e crea un software per consultare i titoli della tua videoteca lontano dal PC

INTERNET

PHP SOTTO STEROIDI

Compila gli script, salvali in una cache e raddoppia la velocità dei tuoi siti

SISTEMA

THREAD? FACILE CON .NET

Scopri il BackGroundWorker e usalo per far lavorare insieme due o più processi

JAVA

UN MONITOR PER LE APPLICAZIONI

Usa JMX per interrogare il tuo software. Funziona anche da remoto

COSTRUIAMO UN FILE DI HELP

Premi il tasto F1 e visualizza l'aiuto in linea, completo di indice e ricerca!

.NET

DATI IN VISTA QUASI COME EXCEL

Arriva la nuova DataGridView di .NET 2.0, ecco come sfruttarne tutte le potenzialità

PROGRAMMA MSN MESSENGER

Integralo nelle applicazioni e usalo per scambiare informazioni in tempo reale

INSTALLAZIONI SEMPLIFICATE

Risolvi ogni problema di upgrade e di distribuzione con ClickOnce
IN PIÙ LA VIDEOGUIDA NEL CD

WORKFLOW FOUNDATION HOWTO

Disegna il comportamento della tua applicazione, il codice arriva in automatico...
IN PIÙ LA VIDEOGUIDA NEL CD

ADDIO PROBLEMI CON I TIPI

Impara ad usare i Generics: una delle novità più importanti del framework 2.0

ALBERI BINARI Dalla ricerca alla ricorsione, ti spieghiamo come funzionano gli algoritmi fondamentali!

EDIZIONI MASTER
www.edmaster.it

Anno X - N.ro 6 (103) - Giugno 2006 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Fabio Farnesi
Collaboratori: M. Autiero, C. Bellucci, M. Bigatti, R. Brunetti, L. Buono, L. Caputo, D. De Michelis, C. Durante, F. Grimaldi, M. Scala, G. Sudano

Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Art Director: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Elio Monaco, Francesco Cospite

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831213 - Fax 02 83121330
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI

ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90 · ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 30/06/06
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ **tel. 02 831212**
@ **e-mail: servizioabbonati@edmaster.it**

Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Maggio 2006

Audio Video Foto Bild, A-Team, Calcio & Scommesse, Colombo, Computer Bild Italia, Computer Games Gold, Digital Japan Magazine, Digital Music, Distretto di polizia, DVD Magazine, Filmteca in DVD, Giochi e Programmi per il tuo telefonino, Go!OnLine Internet Magazine, Guide di Win Magazine, Guide Strategiche di Win Magazine giochi, Home Entertainment, Horror mania, I Corsi di Win Magazine, I Fantastici CD-Rom, I film di idea web, I Filmissimi in DVD, I Libri di Quale Computer, I Mitici all'Italiana, Idea Web, InDVD, IoProgrammo, Japan Cartoon, La mia Barca, La mia Videoteca, Le Grandi Guide di ioProgrammo, Linux Magazine, Magnum PI, Miami Vice in DVD, MPC, Nightmare, Office Magazine, Play Generation, Popeye, PC Junior, PC VideoGuide, Quale Computer, Softline Software World, Supercar in dvd, Thriller Mania, Win Junior, Win Magazine Giochi, Win Magazine, Le Collection, Le Femme Fatale del Cinema.

Direzione TV

Tutti si interrogano: qual è il futuro di Internet? qual è il futuro della programmazione? La verità è che soprattutto Internet emana un fascino fuori dal comune nei confronti di chi la utilizza ed in particolare nei confronti di chi è appassionato di tecnologia. L'evoluzione della rete è chiara, si è passati dalla magnificazione dei contenuti, al boom dell'e-commerce, alla democrazia dei blog. Per ciascuno di questi "strati temporali" si è vissuto almeno un momento di esaltazione comunitaria, in cui la rivoluzione portata della rete avrebbe potuto cambiare le sorti economico/sociali del nostro mondo. E per quanto gli scettici rimangano dubbiosi, è percepibile quanto Internet, intesa come insieme dei suoi servizi, sia ormai presente nella nostra vita quotidiana e quanto giornalmente incida sulle nostre attività. Siamo dunque ad una nuova frontiera, quella della Televisione over IP. Anche questa volta come sempre si grida alla rivoluzione, ed anche questa volta come sempre non resteremo delusi. Certo chi si aspetta che dall'oggi al domani l'audience delle televisioni tradizionali venga intaccato dalla televisione via Internet, rimarrà deluso. La televisione over IP entrerà nelle nostre abitudini di vita, così come ha fatto la posta elettronica, così come ha fatto l'ecommerce, così come continuano a fare i blog, lo farà con discrezione come solo Internet è abituata a fare. Noi programmatori ci troviamo di fronte ad una nuova sfida, dopo avere creato il web dinamico, dopo avere automatizzato la pubblicazione di contenuti testuali, dopo avere inventato i mezzi per consentire a chiunque di avere una voce su Internet, siamo adesso al momento in cui dovremo essere capaci di automatizzare la pubblicazione di Video. Lo faremo, come sempre, stando attenti ai particolari e aiutando la tecnologia a crescere. Lo faremo pensando all'ottimizzazione della banda, alla veicolazione della pubblicità, all'interattività con l'utente, alla raccolta dei log, e infine pensando ai contenuti On Demand e alla loro commercializzazione. Un anno importante ci attende in questo settore, la nuova sfida si gioca sui servizi, e la TV via Internet può essere uno dei centri proprio per l'erogazione di tutti quei servizi che solo noi sappiamo programmare!

Fabio Farnesi ffarnesi@edmaster.it

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre \soft\codice\ e \soft\tools\) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

TELEVISIONE OVER IP

Tutti la chiedono. Ecco le cose da conoscere per non farti trovare impreparato

- ✔ **TEORIA** Mai più dubbi su Tv On Demand, Broadcasting, larghezza di banda e compressione
- ✔ **STRUMENTI** Windows Media SDK, Encoder e Media Services. Impara come usarli!
- ✔ **PRATICA** Crea una trasmissione aggiungi la pubblicità e gestisci lo streaming
- ✔ **CODICE** Un'applicazione completa e personalizzata in Visual Basic

pag. 14

SQL SERVER 2005 USALO COSÌ!

Rendi tutto velocissimo con le stored procedure, automatizza il lavoro con i Jobs, crea DLL in .NET e registrale nel database

IOPROGRAMMO WEB

DATABASE

NETWORKING

SISTEMA

SISTEMA

DATABASE

SISTEMA

SOLUZIONI

RUBRICHE

IOPROGRAMMO by EXAMPLE

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

http://forum.ioprogrammo.it

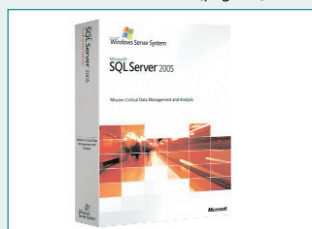
RIVISTA + 2 CD-ROM in edicola

Prodotti del mese

Microsoft SQL 2005 Server Express Edition

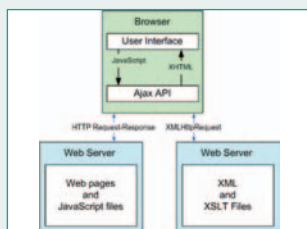
Per la prima volta in versione gratuita, arriva il DB per le aziende

Microsoft SQL server 2005 è la nuova versione del database progettato da Microsoft per rispondere alle esigenze di tipo professionale. Le novità introdotte sono veramente moltissime. Prima fra tutte spicca il nuovo formato file: "MDF". I dati possono essere salvati in un file esterno, con estensione MDF. Questo approccio, molto simile a quello usato da Access con i suoi MDB, rende incredibilmente elevata la portabilità da una macchina all'altra. Ulteriori novità importanti sono state introdotte come ad esempio i Service Broker, che consentono di salvare i dati in una coda del SQL Server locale ed effettuare un update sul server remoto in modo automatico alla prima disponibilità della connettività, e ancora gli Integration Services grazie ai quali è possibile migrare da qualunque database esterno anche proprietario a SQL Server

[pag.106]

Ajax For .NET

Il framework per creare applicazioni AJAX in ambiente Microsoft

Ajax è una tecnologia che si sta rapidamente affermando nel campo delle WEB Application. Consente di realizzare pagine web dinamiche in grado di aggiornare solo le parti variabili della pagina senza doverla ricaricare interamente.
Ovviamente questo comporta un vantaggio immediato in termini di usabilità delle applicazioni, tale che le web application possono assumere il comportamento tipico in termini di interfaccia, di un'applicazione standalone. Grazie a questo genere di possibilità il Web sta rapidamente migrando concentrandosi sull'erogazione di servizi all'utente finale, basati su interfacce che fino a qualche anno fa sembravano impossibili per la rete internet

[pag.107]

AXIS 1.3

Il framework per lo sviluppo di Web Service in Java

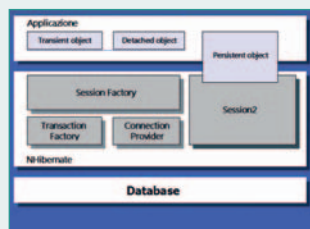
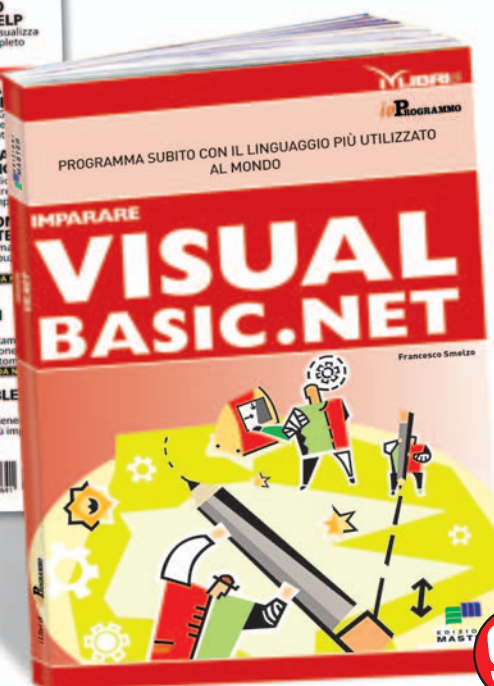
Che i Web Service rappresentino una risorsa enorme per la programmazione è ormai noto, in questo stesso numero di ioProgrammo diamo enorme spazio all'argomento proprio con una serie di esempi pratici per lo sviluppo. Mancano dai nostri esempi proprio quelli relativi a Java di cui ci occuperemo nei numeri successivi della rivista. Axis è comunque il tool fondamentale per poter sviluppare WS con il linguaggio di SUN. Il suo utilizzo è abbastanza semplice, e supporta tutte le caratteristiche avanzate come ad esempio la generazione dinamica dei WSDL oppure il debugging delle trasmissioni SOAP Lavora in congiunzione stretta con Tomcat e il suo deployment si riduce a copiare i file nella directory corretta.. Uno strumento indispensabile di cui non mancheremo di parlare ulteriormente.

[pag.106]

Hibernate 3.2.0

Il tool per la persistenza dei dati in JAVA

I programmatori sono abituati a pensare in termini di classi ed oggetti. Non c'è programma che prescinda ormai dalla logica della programmazione OOP. Tuttavia i database SQL ragionano ancora in termini di puro linguaggio, non c'è nessuna correlazione fra un dato e l'oggetto che lo rappresenta. Hibernate aggira questo ostacolo, ponendosi come framework che maschera un database relazionale con una logica OOP.
In questo modo i programmatori Java possono ad esempio accedere a un qualunque database pensando alle righe e alle colonne che lo rappresentano in termini di oggetti e non di entità relazionali, con evidenti vantaggi in termini di sviluppo omogeneo. Quella che vi presentiamo in questo numero è la versione più aggiornata del tool, più veloce e affidabile.

[pag.108]

RIVISTA + LIBRO + 2 CD-ROM in edicola

I contenuti del libro

Imparare VISUAL BASIC .NET

Visual basic è uno di quei linguaggi che hanno cambiato il modo di intendere la programmazione nel corso del tempo. Fin dalla sua prima uscita, grazie al concetto di RAD e di programmazione ad eventi ha conquistato il cuore di milioni di programmatori in tutto il mondo, tanto che attualmente è sicuramente fra i linguaggi più utilizzati dalle aziende e dai singoli team di sviluppo.
La naturale evoluzione di Visual Basic nel modello .NET segna un ulteriore importante passo nello sviluppo di questo linguaggio. "Imparare VB.NET" si pone come un manuale pratico, che prescinde da una conoscenza approfondita delle versioni precedenti di Visual Basic, e che porta anche l'utente più inesperto, in breve tempo a poter costruire applicazioni complete.

PROGRAMMA SUBITO CON IL LINGUAGGIO PIÙ UTILIZZATO AL MONDO

- Gli elementi di base
- Istruzioni e flussi di codice
- Gli oggetti e le classi
- Il .NET Framework

VISUAL STUDIO 2005
UN BUON MOTIVO PER SVILUPPARE CON TECNOLOGIA .NET

MICROSOFT HA RILASCIATO LO SCORSO NOVEMBRE LA NUOVA VERSIONE 2005 DI VISUAL STUDIO, LO STRUMENTO DI SVILUPPO PIÙ COMPLETO PER REALIZZARE APPLICAZIONI PER WINDOWS, PER IL WEB, WEB SERVICE E DISPOSITIVI MOBILI

Dal 1° marzo 2006, il prodotto è disponibile per il mercato anche in lingua italiana: interfaccia utente (menù e barra degli strumenti), documentazione (MSDN Library), starter kit, descrizione delle classi e degli errori di codice, ora totalmente nella nostra lingua, rendono Visual Studio 2005 ancora più intuitivo e facile da usare. Oltre 400 le nuove funzionalità rispetto alle precedenti versioni .NET 2002 e 2003, che rendono Visual Studio 2005 decisamente più produttivo.

PRODUTTIVITÀ

Questa è la parola chiave su cui si fonda questa nuova versione dell'ambiente di sviluppo ideato da Microsoft per la programmazione .NET.
Se da un lato si registra un'alta integrazione con la versione 2.0 del .NET Framework, dall'altro non possono passare inosservati wizard e accorgimenti di vario genere che caratterizzano il nuovo ambiente di sviluppo rispetto alle versioni precedenti.

FRAMEWORK 2.0

L'intero IDE si basa sul Framework 2.0 e ne sfrutta a fondo tutte le potenzialità. Non è più possibile sviluppare per il Framework 1.1, nè consigliabile. Il nuovo Framework porta con sé novità rilevanti quali i tipi generics, nuovi controlli, un maggior numero di classi, ed in generale una maggiore efficacia. Visual Studio 2005 è altamente integrato con questa versione del Framework e ne diventa lo strumento naturale per lo sviluppo in tecnologia .NET.

DATABASE

Tutti sanno quanto lo sviluppo delle applicazioni sia spesso legato all'utilizzo di una base di dati. In questo senso le facilities messe a disposizione da Visual Studio 2005 sono veramente straordinarie. Il cuore di tutto è un wizard per l'aggiunta di una sorgente dati. Il wizard riesce a realizzare una stretta connessione, sia con fonti di database tradizionali quali SQL Server 2005, Access o qualunque altra fonte per cui esista un provider, sia con Web Services e persino con una classe sviluppata ad hoc dallo sviluppatore per la gestione di connessioni proprietarie. Quel che più conta, è che da questo momento in poi la gestione del database diventa completamente trasparente per l'utente. È sufficiente trascinare un campo del db dal database explorer sulla form per poterne gestire la visualizzazione ed eventualmente l'inserimento dati. Una semplice applicazione per la gestione di un database si realizza in meno di cinque minuti, specialmente se la fonte è un db SQL Server 2005 oppure Access. Da segnalare anche l'arrivo di una nuova DataGridView che sostituisce la vecchia DataGrid e porta con se una ventata di novità specialmente per quanto riguarda la personalizzazione. A margine di tutto questo è da segnalare che SQL Server 2005 gestisce adesso la base di dati con file esterni con estensione .mdf, con una logica molto simile a quella del vecchio mdb di Access, il che rende facilmente portabile da una macchina all'altra qualunque database.

DATABASE DESIGNER

In Visual Studio 2005 è presente uno strumento per la gestione delle relazioni, delle query e in generale per la progettazione delle basi di dati. Anche in questo caso si tratta di uno strumento particolarmente efficace e legato al concetto di integrazione.

SMART TAG

Altra novità molto interessante di Visual Studio 2005 sono gli Smart Tag, una sorta di menù contestuale dinamico e agganciato a controlli specifici, in grado di mostrare le funzionalità più comuni. Così, per esempio, tramite lo smart tag della DataGridView è possibile accedere direttamente ad un Wizard per la definizione delle colonne, oppure per il binding dei contenuti.

FORM DESIGNER

Sembra una novità di poco rilievo, eppure anche in questi particolari è importante notare come si sia posta grande attenzione alla produttività. I controlli possono essere adesso allineati fra di loro, in modo molto semplice, utilizzando delle linee guida calamitate.

CLICKONCE

Si tratta di un framework estremamente comodo, nato per risolvere i problemi della distribuzione del software, che permette all'utente d'installare in locale o usare da remoto le applicazioni per la piattaforma Windows 2000/2003/XP 2006. Tramite ClickOnce, l'utente finale sarà in grado di utilizzare un'applicazione Windows Forms (cioè un'applicazione .NET per Windows) con un solo click del mouse. A seconda della tipologia di distribuzione che s'intende affrontare, sarà possibile scegliere una modalità d'installazione dal web, oppure standalone. In tutti i casi, grazie a ClickOnce,

eventuali disponibilità di nuove versioni saranno comunicate all'utente e l'upgrade sarà estremamente facilitato.

ASP .NET 2.0

Moltissime novità sono state introdotte nei tool relativi allo sviluppo Web. Template e Pagine Master ne sono un esempio. È adesso possibile definire una pagina master, al cui interno possono essere contenute parti variabili e parti fisse. Ogni pagina ereditata da una pagina master potrà personalizzare le parti variabili, ma rimarrà omogenea alla pagina originaria per quanto riguarda le parti fisse. Allo stesso modo i template consentono di generare una grafica generale per il sito. Cambiando il template si cambierà l'intero layout del progetto senza dover ridisegnare tutte le pagine. Non di poco conto l'inserimento, all'interno dell'ambiente di sviluppo, di un Web Server per il testing delle pagine. Nella versione precedente era necessario installare sulla propria macchina un server IIS per poter testare il software. Con Visual Studio 2005 è sufficiente lanciare l'applicazione e contestualmente verrà lanciato un server web compatibile con ASP .NET 2.0 su una porta random che eseguirà il codice sviluppato.

MOBILE EDITION

Con Visual Studio 2005, eccetto che nelle versioni Express, è possibile anche sviluppare applicazioni per dispositivi mobili. Il porting di un'applicazione standalone ad una progettata per un apparecchio mobile si riduce il più delle volte semplicemente a riprogettarne l'interfaccia utente.

COMPATIBILITÀ CON LE VERSIONI PRECEDENTI

Il porting da Visual Studio 2003 è assolutamente indolore. Un comodo Wizard eseguirà le operazioni necessarie perché tutto funzioni nel nuovo ambiente. Leggermente più complesso il porting da Visual Basic 6.0, ma Visual Studio 2005 ha colmato tutte le carenze della precedente versione, divenendo uno strumento estremamente più potente e versatileIn qualche occasione sarà necessario modificare manualmente il codice perché tutto funzioni correttamente. Tuttavia, c'è da dire che VB 6.0 non è più ufficialmente supportato da Microsoft, per cui anche i più restii dovranno abituarsi all'idea di un upgrade al nuovo prodotto.

SOLUZIONI SU MISURA

Visual Studio 2005 viene proposto in diverse edizioni, tagliate su misura per le esigenze dei singoli programmatori:

VISUAL STUDIO 2005 TEAM SYSTEM

Strumenti produttivi, integrati ed estensibili per la **gestione del ciclo di vita del software**, pensati appositamente per ottimizzare le comunicazioni e la collaborazione tra i team durante l'intero processo di sviluppo.

VISUAL STUDIO 2005 PROFESSIONAL EDITION

Si tratta di una versione pensata per il singolo sviluppatore, o per piccoli gruppi di sviluppo. Contiene tutti gli strumenti per realizzare applicazioni mission-critical, smart client, Web, mobili o basate su Microsoft Office. Permette di accedere in modo completo all'ambiente di sviluppo Microsoft .NET Framework 2.0, e supporta la creazione di strumenti che estendono le funzionalità dell'ambiente IDE (Integrated Development Environment) di Visual Studio. Sono inoltre disponibili caratteristiche migliorate come "Edit and Continue", che consente di individuare e correggere rapidamente gli errori senza riavviare le applicazioni.

VISUAL STUDIO 2005 STANDARD EDITION

Strumenti di sviluppo volti alla **creazione di applicazioni client/server** e siti Web. Per gli sviluppatori che passano da Visual Basic 6.0 a Microsoft .NET Framework 2.0, Visual Studio 2005 Standard Edition **migliora la produttività**, e offre la potenza necessaria per la creazione di applicazioni line-of-business.

VISUAL STUDIO 2005 EXPRESS EDITION

Si tratta di prodotti estremamente intuitivi e di facile utilizzo, dedicati in particolare a utenti alle prime armi, completamente gratuiti e disponibili per il download e relativa registrazione sul sito MSDN (http://www.microsoft.com/italy/msdn/default.mspx). Si tratta di una licenza completa, liberamente utilizzabile anche in fase di produzione a tempo indeterminato, a cui mancano però strumenti avanzati di reportistica, funzionalità per sviluppo Mobile e altre caratteristiche disponibili nelle altre edizioni per professionisti.
Visual Studio 2005 è disponibile sia come Licenza singola (Visual Studio 2005 nelle sue diverse edizioni), sia come Licenza + Software Assurance (Visual Studio 2005 nelle sue diverse edizioni con MSDN Professional o con MSDN Premium). La Software Assurance di Visual Studio 2005 è MSDN, ossia la sottoscrizione periodica che garantisce il diritto a ricevere aggiornameni gratuiti alle ultime versioni, beta di prodotto in anteprima, software downloadabile gratuitamente, supporto tecnico gratuito e molti altri vantaggi.

I prezzi di Visual Studio 2005 nelle sue diverse edizioni variano a seconda delle modalità di acquisto:

- **modalità Retail**, ossia come prodotto pacchettizzato (la scatola del prodotto), per chi necessita di una singola licenza;
- **modalità Volume Licensing** (multilicenza), acquistabile nell'ambito di uno dei seguenti programmi: Easy Open, Open Volume, Open Value, Select, Enterprise Agreement ed Enterprise Agreement Subscription.

CONCLUSIONI

Visual Studio 2005 rappresenta un elemento di rottura rispetto alle versioni precedenti della piattaforma di sviluppo, grazie alla presenza di una serie di funzionalità che migliorano notevolmente la produttività individuale. Inoltre, la disponibilità delle versioni Express Edition, rappresenta un'apertura anche verso il mondo del software OpenSource, che può in questo modo iniziare a sviluppare applicazioni .NET anche per l'ambiente Windows senza ricorrere a impegni monetari importanti.

GLI ALLEGATI DI IOPROGRAMMO

▼ Eleganza e potenza

Fino a qualche tempo fa il linguaggio di punta promosso da Microsoft era Visual Basic, ed è ancora così... tuttavia da qualche tempo a questo monolite della programmazione si è affiancato C# che, per eleganza, disponibilità di costrutti, capacità di programmazione a basso livello è diventato uno dei più amati dai programmatori professionali. Il linguaggio, da molti inizialmente criticato perché troppo simile a Java, è diventato nel tempo un punto di riferimento sicuro ed un'irrinunciabile per chiunque si appresti a programmare applicazioni in ambito Windows. In realtà come ben si può intuire, si presta molto bene anche alla costruzione di applicazioni web e per questo motivo assume quei connotati di flessibilità che lo rendono così amato dai programmatori di tutto il mondo. I tre WebCast che ioProgrammo in collaborazione con MSDN vi presenta questo mese vi porteranno proprio a conoscere le basi per iniziare a programmare in C#. Si tratta di un viaggio affascinante attraverso alcuni dei costrutti più eleganti disponibili in programmazione.

I VIDEOCORSI PER PROGRAMMARE BENE

3 WEBCAST UFFICIALI MICROSOFT

IN ESCLUSIVA GRATIS NEL CD "I CORSI DI FORMAZIONE" DA SEGUIRE COMODAMENTE SUL PC

msdn WEBCAST

VISUAL C# 2005

- Introduzione a C#
- C# Advanced Edition (prima parte)
- C# Advanced Edition (seconda parte)

INFORMAZIONI SU MSDN WEBCAST
http://www.microsoft.it/msdn/webcast_msdn
http://forum.ioprogrammo.it

FAQ

Cosa sono i Webcast MSDN?

MSDN propone agli sviluppatori una serie di eventi gratuiti online e interattivi che approfondiscono le principali tematiche relative allo sviluppo di applicazioni su tecnologia Microsoft. Questa serie di "corsi" sono noti con il nome di Webcast MSDN

Come è composto tipicamente un Webcast?

Normalmente vengono illustrate una serie di Slide commentate da un relatore. A supporto di queste presentazioni vengono inserite delle Demo in presa diretta che mostrano dal vivo come usare gli strumenti oggetto del Webcast

Come mai trovo riferimenti a chat o a strumenti che non ho disponibili nei WebCast allegati alla rivista?

La natura dei WebCast è quella di essere seguiti OnLine in tempo reale. Durante queste presentazioni in diretta vengono utilizzati strumenti molto simili a quelli della formazione a distanza. In questa ottica è possibile porre domande in presa diretta al relatore oppure partecipare a sondaggi etc. I WebCast riprodotti nel CD di ioProgrammo, pur non perdendo nessun contenuto informativo, per la natura asincrona del supporto non possono godere dell'interazione diretta con il relatore.

Come mai trovo i WebCast su ioProgrammo

Come sempre ioProgrammo cerca di fornire un servizio ai programmatori italiani. Abbiamo pensato che poter usufruire dei WebCast MSDN direttamente da CD rappresentasse un ottimo modo di formarsi comodamente a casa e nei tempi desiderati. Lo scopo tanto di ioProgrammo, quanto di Microsoft è infatti quello di supportare la comunità dei programmatori italiani con tutti gli strumenti possibili.

Su ioProgrammo troverò tutti WebCast di Microsoft?

Ne troverai sicuramente una buona parte, tuttavia per loro natura i webcast di Microsoft vengono diffusi anche OnLine e possono essere seguiti previa iscrizione. L'indirizzo per saperne di più è: http://www.microsoft.it/msdn/webcast/msdn segnalo nei tuoi bookmark. Non puoi mancare.

L'iniziativa sarà ripetuta sui prossimi numeri?

Sicuramente si.

News

DISPONIBILE MOSAICO SORGENTE APERTO 8.0

Whag, la software house produttrice del noto gestionale Mosaico ne ha appena rilasciato la versione 8.0. Rispetto alle precedenti versioni sono da annotare la presenza del “sensore di allarme sotto scorta”, di una funzionalità “cruscotto” dedicata a chi vuole monitorare il fatturato aziendale e SQLExec che consente di eseguire delle query dirette sul DB di Mosaico. Rimane invariata la disponibilità del codice sorgente. Da sempre Whag ha infatti adottato la politica di rendere disponibile il codice per gli sviluppatori affinché possano eventualmente apportare le modifiche necessarie per le esigenze delle aziende, che come si sa sono talmente tante ed eterogenee che è quasi impossibile trovare un gestionale univoco per ogni tipologia di problema. La disponibilità del codice sorgente è sinonimo di personalizzazione e flessibilità, decisamente un punto a favore di Whag e di Mosaico.

PHP APPRODA AL SUMMER OF CODE 2006

Con una delle sue grandi trovate Google, nel 2005, aveva inaugurato la stagione di Summer of Code. Sostanzialmente una borsa di studio con cui software developer meritevoli guadagnavano una esperienza all'interno di organizzazioni OpenSource nel campo dello sviluppo. Ciascuno studente riceveva uno stipendio e la possibilità di contribuire al miglioramento di un progetto OpenSource portando le sue idee e il suo lavoro all'interno dell'organizzazione che ne curava lo sviluppo. L'idea ha avuto un enorme successo, tanto che l'operazione continua con la versione 2006 del Summer of Code. Le organizzazioni partecipanti sono davvero moltissime e i nomi altisonanti, fra tutti abbiamo scelto PHP che quest'anno apre agli sviluppatori ricercando, attraverso Summer of Code, profili in grado di migliorare la nota piattaforma per lo sviluppo di web application. Nella home page del progetto: http://code.google.com/summerofcode.html, campeggiano comunque nomi altisonanti, da OpenSolaris ad Eclipse, da PostgreSQL a Samba.

IN ARRIVO LA JAVA CONFERENCE 2006

Si terrà a Roma il 26 Giugno 2006, e a Milano il 27 e il 28 Giugno l'undicesima Java Conference. Come di consueto l'evento è rivolto a sviluppatori e a professionisti dell'IT in generale, ma diversamente dagli anni precedenti alle consuete giornate Milanesi se ne è aggiunta una Romana che testimonia la buona vitalità del mercato dello sviluppo anche nelle regioni del centro sud.
Il tema di quest'anno sembra ripercorrere uno dei grandi interrogativi di tutti coloro che in un modo o nell'altro hanno investito nel settore dell'informatica: “Il futuro di Internet, il futuro del software”.
Al centro di ogni discussione l'evoluzione del Web, che con la crescente disponibilità di banda e l'aumentare dei servizi disponibili influenza l'evoluzione di un nuovo modello di società, ben rappresentata dal termine “Social Networking”, ovvero un'insieme di persone che in nome di interessi comuni formano un nucleo ben definito a cui Internet presta la propria infrastruttura. Non a caso Sun ha battezzato “Partecipation Age” un'epoca in cui attraverso la rete individui e operatori interagi-

IL PENTAGONO ATTENDE GARY MCKINNON

L'hacker noto per avere forzato i sistemi informatici della Nasa è attualmente in attesa di conoscere il proprio destino. Se i giudici Inglesi dovessero concederne l'estradizione in America, l'uomo rischierebbe fino a 70 anni di carcere!
Il trentanovenne Londinese, Gary McKinnon, noto in rete con il nickname di Solo era stato arrestato per la prima volta nel 2002 con l'accusa di avere violato circa 97 computer appartenenti al dipartimento della difesa americano, alla marina e all'esercito. I danni subiti dai vari network sotto attacco ammonterebbero secondo una stima a 900.000 dollari. L'uomo era stato immediatamente rilasciato nel 2002 per poi essere arrestato una seconda volta nel 2005. Anche il secondo arresto aveva dato esito negativo, tuttavia il rilascio, questa volta, era stato subordinato al divieto per l'uomo ad accedere alla rete Internet. Se l'estradizione fosse confermata, si prevede che gli Americani non sarebbero altrettanto teneri, e che per McKinnon si aprirebbero le porte di Guantanamano. Nel frattempo “Solo” si difende dicendo stava semplicemente cercando di capire se gli UFO esistono e afferma di essere sicuro che gli americani stanno lavorando su tecnologie “Antigravitazionali” che modificherebbero sostanzialmente il modo di affrontare le missioni nello spazio

sono continuamente per creare valore, opportunità e crescita.
In questo contesto si pongono gli strumenti di Sun che sono concepiti secondo modelli di sviluppo innovativi tesi, da un lato a servire la crescente necessità di interazione attraverso la rete, dall'altro a garantire una maggiore sicurezza nell'utilizzo del networking e dei sistemi. Le parole d'ordine sembrano essere dunque: "Software inteso come servizio", "Open Source e architetture orientate ai servizi" e infine ma non ultimo in ordine di importanza: "Ajax". Fra le presenze importanti, da segnalare quella di James Gosling, padre fondatore di Java e attualmente Vice Presidente di Sun Fellow Sun Microsystems e quella si Glenn Edens responsabile di Sun Labs.
Le informazioni sulla registrazione e gli aggiornamenti sul programma sono disponibili su:
www.sun.it/eventi /jc06

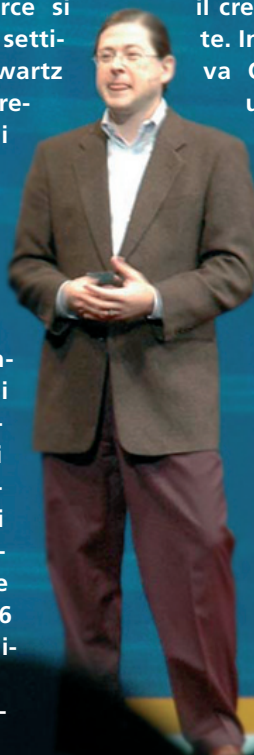
È ANCORA GUERRA FRA GOOGLE E MICROSOFT

L'oggetto della contesa è questa volta il nuovo Internet Explorer 7.0. Ancora non disponibile sul mercato, il nuovo prodotto è già portatore del seme della discordia. Esattamente come nel caso di Firefox il nuovo browser integrerebbe una funzionalità di ricerca direttamente collegata a MSN Search di Microsoft che come si sa è un diretto concorrente di Google. Dato l'alto numero di installazioni del browser, appare ovvio che questo tipo di strumento integrato si configurerebbe come una minaccia al libero mercato. E' interessante notare come la lotta si sia spostata dall'integrazione di Explorer all'interno del sistema operativo all'integrazione di funzioni particolari all'interno del browser. Il concetto è sempre lo stesso. Chi detiene quote così enormi di mercato non dovrebbe integrare nei propri strumenti di punta servizi all in one tesi a condizionare in qualche modo gli utenti nella propria volontà all'acquisto o all'utilizzo di un servizio. Almeno questa è la tesi degli avversari di Microsoft. Assisteremo ad un'altra lotta come quella condotta per estrapolare il browser dal sistema operativo? In quell'epica battaglia a farne le spese fu Netscape, riuscirà il gigante Google a resistere all'avanzata del colosso di Redmond? Questa volta tuttavia le parti sembrerebbero invertite, di fatto il leader nel campo dei motori di ricerca è Google, mentre Microsoft se pur utilizzando mezzi discutibili cerca di ritagliarsi una fetta di nella raccolta pubblicitaria sul Web, attualmente appannaggio quasi esclusivo proprio del rivale Google.

ARRIVA MICROSOFT EXPERIMENTATION PLATFORM

Volete sviluppare un nuovo servizio per Windows Live? potete farlo adesso effettuando le vostre sperimentazioni in un ambiente appositamente studiato per il testing. La piattaforma di sperimentazione è disponibile all'indirizzo *http://exp-platform.com /default. aspx* e consente di lavorare in tutta tranquillità per estendere Windows Live: *http://www .live.com*. Windows Live è l'ennesimo test effettuato da Microsoft per garantire un'esperienza di navigazione e ricerca personalizzata per un utente. Il servizio, ancora in beta, assomiglia, neanche a farlo apposta, proprio a Google, ma consente di personalizzare la propria pagina aggiungendovi contenuti come Feed Rss o altri elementi dinamici. La creazione di questo "Labs" dedicato agli sviluppatori lascia intendere come nei prossimi anni, il terreno di scontro si sposterà sui servizi integrati, e come il concetto di sito web si sposterà per diventare "Servizio" web. L'acquisizione di nuovi utenti corrisponderà necessariamente anche a una maggiore capacità di attrarre investitori pubblicitari e tutto questo si potrà concretizzare solo offrendo servizi a valore aggiunto che svincolino l'utente da una piattaforma specifica.

JAVA SARÀ OPENSOURCE?

Rumors relativi a dichiarazioni di Johnatan Schwartz sulla possibilità di rendere Java completamente OpenSource si sono diffusi nelle ultime settimane. La volontà di Schwartz di procedere in questa direzione sarebbe conferma di quanto Sun stia esplorando ogni possibilità pur di tornare a produrre utili consistenti.
Di fatto nonostante il linguaggio sia ormai il leader indiscusso all'interno di dispositivi mobili e nonostante l'ottima diffusione anche in ambiti di programmazione tradizionale, tuttavia gli analisti rimangono scettici di fronte al bilancio di Sun che nel terzo quarto del 2006 avrebbe perso ben 217 milioni di dollari.
Shwartz rimane invece ottimista. Gli investimenti fin qui effettuati saranno capitalizzati negli anni avvenire e andranno di pari passo con il crescere dei servizi di rete. In ogni caso rendere Java Open Source sarebbe un passo importante, che da un lato incrementerebbe ancora di più la già enorme base di utenti, ma dall'altro potrebbe dar vita ad una serie di dialetti che non favorirebbero uno sviluppo omogeneo delle applicazioni, della documentazione e del segmento dei servizi. La tensione rimane alta, e molto di più ne sapremo proprio durante l'ormai imminente Java Conference 2006.

LA TELEVISIONE VA SU INTERNET

BROADCASTING, UNICASTING, TV ON DEMAND, PAROLE CHE STANNO ENTRANDO NEL GERGO COMUNE. ECCO LE COSE CHE UN PROGRAMMATORE DEVE CONOSCERE PER FORNIRE RISPOSTE ADEGUATE AI PROPRI CLIENTI

In questo articolo: mostreremo sia le tecnologie messe a disposizione da Microsoft per la trasmissione di audio e video su protocollo TCP/IP, sia come noi programmatori possiamo utilizzare queste conoscenze per creare applicazioni in grado di proiettare contenuti multimediali su Internet. Lo scenario è il seguente:

1) Il nostro cliente vuole creare una televisione su Internet

2) Desidera sia offrire contenuti in diretta televisiva, sia offrire contenuti on demand

3) Non vuole perdere tempo con la tecnologia. Ciò che ci chiede è poter premere due tasti e inviare i contenuti nel modo appropriato, senza doversi preoccupare di avere conoscenze approfondite di informatica. D'altra parte il suo mestiere è quello di essere un fornitore di contenuti, non un tecnico.

Offriremo una soluzione basata su Windows 2003, Windows Media Services, e Windows Media Encoder. Nella prima parte di questo articolo illustreremo una parte puramente sistemistica che ci farà capire qual è l'architettura di rete che sfrutteremo. Nella seconda parte creeremo l'applicazione che il nostro cliente ci ha richiesto.

REQUISITI

Conoscenze richieste
Basi di .NET

Software
Windows 2000/XP.
Visual Basic .NET 2003.
Sql Server 2005

Impegno

Tempo di realizzazione

WINDOWS MEDIA SERVICES

WMS è un servizio installabile gratuitamente su Windows 2003. Coloro i quali dispongono di un sistema Windows 2003 possono installare WMS da *Pannello di Controllo/Aggiungi Rimuovi Applicazioni/Aggiungi Rimuovi Componenti di Windows*. Il naturale complemento di WMS è il Windows Media Player 10.
Un utente che volesse visualizzare contenuti video provenienti da un server di streaming WMS dovrebbe utilizzare il protocollo MMS. Per un normale utente sarebbe sufficiente digitare in Windows Media Player 10: *mms://indirizzodelserver/indirizzodipubblicazione/<opzionale indirizzo del video>* per connettersi al server in questione e poter visualizzare il file. Qual è il vantaggio di utilizzare un servizio di streaming rispetto a consentire ad un utente di scaricare completamente un file per visualizzarlo in un secondo momento? Prima di tutto un server di streaming consente la diretta, o come si dice in gergo il *"Live Streaming"*. Nel caso in cui invece si volessero mettere a disposizione contenuti registrati in precedenza, l'utente sarebbe sempre costretto a visualizzarli sul nostro sito, con evidenti vantaggi in termini di *"brand"*.
In sostanza, siamo sicuri che un video o un file MP3 non possa essere copiato, viceversa possa essere semplicemente riprodotto con le modalità che noi riteniamo più opportune.
Potremmo anche decidere di proteggere i contenuti con una password e fornire l'accesso solo agli utenti registrati al servizio. Infine, un server di streaming consente di gestire la pubblicità, il palinsesto, e la banda. Ovvero, di uno stesso file potrebbe fornire una trasmissione a bande diverse a seconda della disponibilità dell'utente che sta accedendo al servizio.

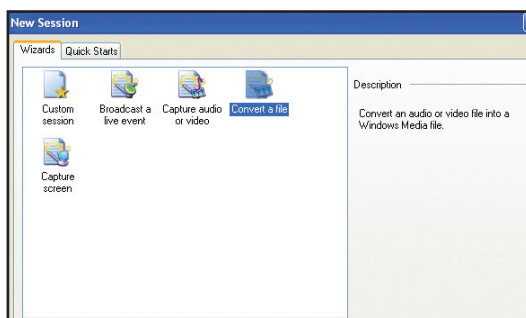
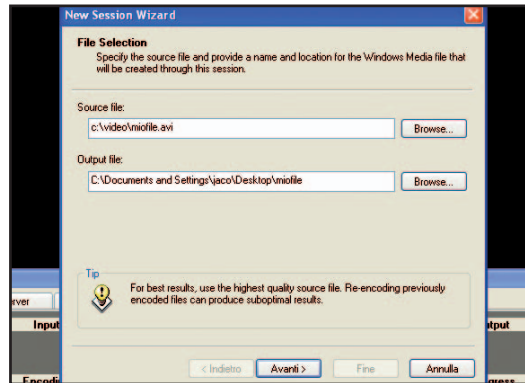
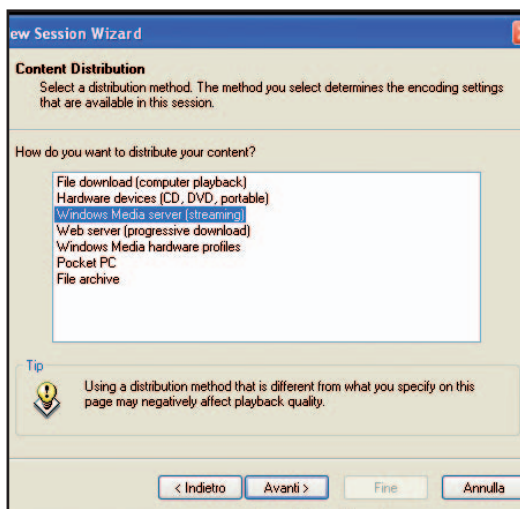
WINDOWS MEDIA ENCODER

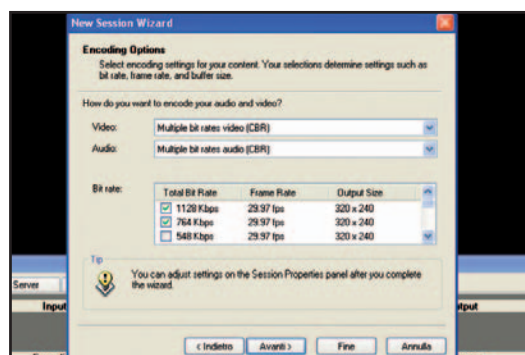
WME è lo strumento di Microsoft che consente di effettuare una codifica audio o video dei dati che si intendono trasmettere. Normalmente per codifica si intende: "compressione". Attualmente siamo giunti alla versione numero 9, e il prodotto è liberamente scaricabile dall'indirizzo *http://www.microsoft.com/windows/windowsmedia/forpros/encoder/default.mspx*. L'importanza di WME all'interno della nostra architettura è fondamentale. Di fatto WME consente di catturare audio e video da una fonte esterna, comprimere i dati in tempo reale e inviarli a un Win-

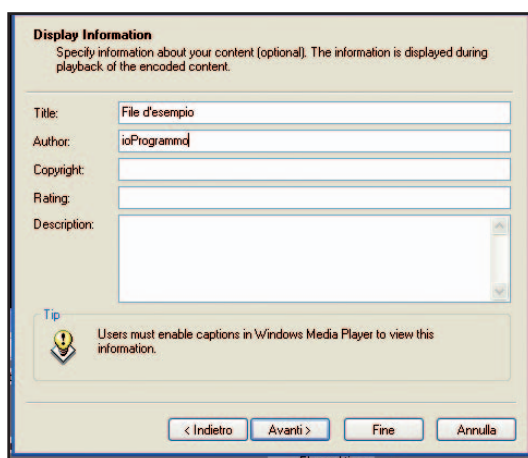
dows Media Services che si occuperà di renderli disponibili a tutti i client che ne vorranno fare uso. Per cui, supponendo che al vostro computer sia collegata una telecamera, Windows Media Encoder, catturerà il segnale dalla telecamera e lo sparerà verso il Windows Media Services. A questo punto un qualunque utente dotato di Windows Media Player potrà connettersi al servizio e visualizzare in tempo reale i dati trasmessi dalla telecamera. In un mondo ideale tutto questo sarebbe già sufficiente a risolvere il nostro problema. Dovremmo installare Windows Media Encoder sul computer del cliente, insegnarli ad usarlo, e tutto sarebbe risolto. In realtà però il WME è un software complesso, che consente diverse modalità di codifica, specializzato in alcune applicazioni, ma mancante completamente di altre. Nessun cliente al mondo vorrebbe usare direttamente il WME, piuttosto desidererebbe un'applicazione semplice che gli consenta di astrarsi totalmente dalla conoscenza sia del software sia dell'architettura sottostante. Per tale motivo è nato il Windows Media Encoder SDK che installa, in una macchina in produzione, tutto l'occorrente per richiamare API messe a disposizione dal WME. Sarà proprio questo componente COM che utilizzeremo all'interno della nostra applicazione .NET per garantire alcune funzioni ed eliminare completamente le altre. Nel frattempo però sarà utile conoscere alcune caratteristiche del Windows Media Encoder che ci serviranno per meglio comprendere in un secondo momento quanto andremo a fare.

LA PRIMA COMPRESSIONE

Inizieremo con il comprimere un file avi già presente sull'hard disk. Non prenderemo per ora dati da una sorgente esterna. Tutto quello che vogliamo fare è familiarizzare con le tecniche di compressione.

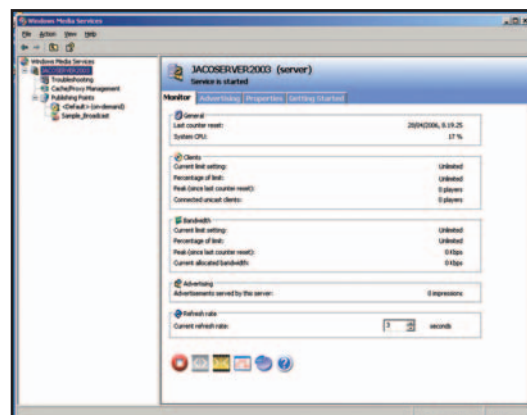
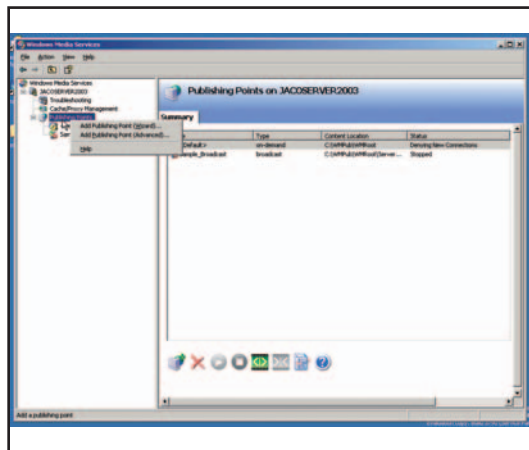
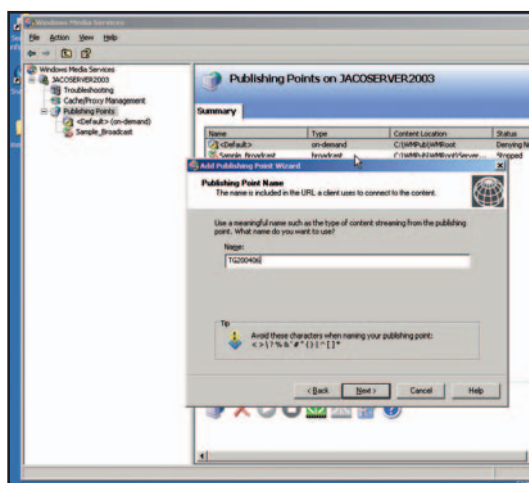
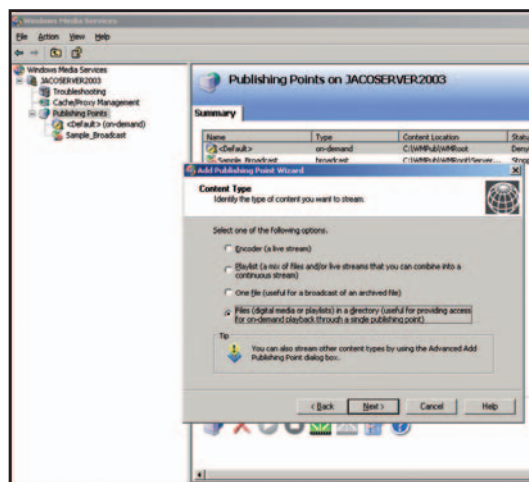
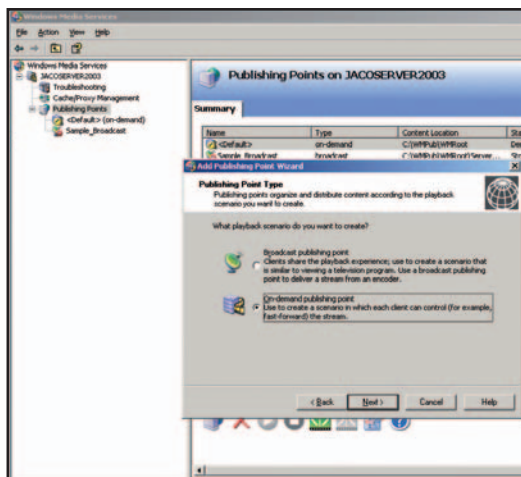
1 Avviamo il Windows Media Encoder e selezioniamo dalla relativa dialog box il progetto *"convert a file"*

2 Selezioniamo il file di input e indichiamo un percorso per il file di output

3 Stabiliamo che il formato della distribuzione debba essere un Windows Media Server, ovvero un server di streaming

4 Stabiliamo la qualità della compressione. Questo passaggio è molto importante. Scegliendo infatti *Multiple Bit Audio* e *Multiple Bit Video*, si produrrà un formato compresso in grado di negoziare la qualità della trasmissione a seconda della banda disponibile, in modo del tutto automatico e senza l'intervento esterno dell'utente. In questo modo chi utilizza un modem per la visualizza-

zione dello stream riceverà un filmato in una qualità tale da essere gestibile dalla propria connessione, chi utilizzerà l'ADSL beneficierà di una qualità senza dubbio più elevata. Da questa finestra dipende molto della qualità del servizio che offriremo ai nostri clienti.

5 Inseriamo informazioni aggiuntive che verranno visualizzate nel player durante la riproduzione del video

Display Information
Specify information about your content (optional). The information is displayed during playback of the encoded content.
Title: File d'esempio
Author: ioProgrammo
Copyright:
Rating:
Description:
Tip
Users must enable captions in Windows Media Player to view this information.
< Indietro | Avanti > | Fine | Annulla

6 Le operazioni di codifica sono terminate. Ora possiamo dare il via alla conversione. Se tutto è andato a buon fine visualizzeremo nel player il risultato del nostro video.

LA PRIMA TRASMISSIONE INTERNET

Per questo nostro primo "tentativo" eviteremo di trasmettere in diretta, piuttosto, renderemo disponibile una trasmissione preregistrata in modalità *"On Demand"*. La TV On Demand è uno dei fenomeni su cui si dibatte negli ultimi anni, e qualcuno fra digitale terrestre e TV satellitare non tiene nella dovuta considerazione la *TV Over IP*. Tv Over IP significa visualizzare sul proprio televisore di casa una trasmissione televisiva, scegliendola quando e come lo si vuole da una televisione basata su Internet come quella che stiamo sviluppando noi. Certo siamo ancora lontani da questo obiettivo, la banda non è sufficiente e le TV moderne non sono ancora in grado di scaricare con una qualità accettabile i flussi Internet. Tuttavia si intravede già quale potrebbe essere la direzione del futuro.
Intanto facciamo qualche distinzione fra: *singolo filmato, trasmissione televisiva, emittente televisiva.*
Per *singolo filmato* si intende quello che comunemente viene indicato come "Servizio", per *trasmissione televisiva* si intende un insieme di servizi, per *emittente televisiva* si intende un soggetto che propone una serie di trasmissioni televisive spesso organizzate in palinsesti.
Ad esempio il classico servizio di un telegiornale è un filmato singolo, un telegiornale è una trasmissione televisiva, la Rai è un'emittente TV. Il nostro scopo sarà proprio riproporre una struttura simile, con la differenza che l'emittente TV se vuole può evitare di creare un palinsesto ad orario, in quanto la nostra TV è On Demand. Per questo motivo, il nostro server equipaggiato con Windows Media Services sarà equiparato a un'emittente TV e corrisponderà al nome del computer o all'indirizzo Internet su cui risiede il servizio. Nel nostro caso utilizzeremo il classico localhost visto e considerato che stiamo facendo degli esperimenti in locale. Indicheremo una "trasmissione televisiva" come *"Publishing Points"*. Ciascun Publishing Points sarà composto da uno o più filmati. I Publishing Points potranno trasmettere in modalità *Broadcasting* o *Unicasting*. Per modalità *Broadcasting* si intende che il video verrà trasmesso sequenzialmente e tutti i client che si connetteranno per guardarlo lo visualizzeranno in maniera sincrona. Ovvero ciascun client vedrà scorrere le stesse immagini nello stesso tempo. Questa modalità è utilizzata soprattutto nelle trasmissione Live in presa diretta, per ovvi motivi. La modalità *Unicast* invece consente a ciascun client di avviare la ripresa del video in modalità asincrona, scegliendo quando e come avviare il video indipendentemente dagli altri. Questa modalità è indicata per la TV On Demand che è quella che utilizzeremo in questo esempio. Infine ciascun *"Publishing Points"* può avere delle caratteristiche. Ad esempio una sigla iniziale e una sigla finale, che devono essere sempre visualizzate qualunque sia il servizio che viene richiesto dall'utente. Oppure pubblicità agganciata ad un particolare servizio sotto la forma di "Banner". Di tutte queste operazioni si occupa il Windows Media Server.

1 Perciò dal menu *start/all programs/administrative tools* avviate *"Windows Media Services"*. Vi comparirà qualcosa di molto simile all'immagine seguente:

2 Cliccate su *"Publishing Points"* e con il tasto destro del mouse su *"Add Publishing Points"*

3 date un nome significativo al vostro Publishing Points, ad esempio *TG200406,* che indica telegiornale del 20 Aprile 2006.

4 selezionate il tipo di trasmissione che volete effettuare, nel nostro caso *"Files in a directory...."*

5 selezionate *On Demand Publishing Point* e la directory dove conserverete i singoli servizi componenti la trasmissione televisiva. Curatevi anche di selezionare il checkbox: *"Enable Access to directory content...."*

6 abilitate il logging se lo volete e proseguite fino alla fine, nella schermata successiva indichiamo di voler creare un:

- *wrapper playlist (wsx)*
- *announcement files (asx)*
- *pagina web (htm)*

vedremo in seguito a cosa ci serve tutta questa roba. Nel frattempo occupiamoci di dire al wizard di creare i file che ci servono e di collocarli in posizioni che gli indicheremo. Possiamo lasciare le impostazioni di default per tutte le schermate che ci verranno proposte.
Il wizard si occuperà di creare per noi i file richiesti. Copiamo all'interno della directory che abbiamo indicato per il publishing points un qualunque file WMV contenente un filmato, e testiamo se tutto funziona collegandoci con il windows media player all'url: *mms://localhost/TG200406.*

LA PLAYLIST

Nel tutorial precedente abbiamo lasciato che il wizard creasse il file *Wrapper Playlist* con estensione *wsx*. Questo file contiene una lista dei servizi che comporranno il nostro publishing points. Se apriamo il file in questione scopriremo che contiene qualcosa del genere:

```
<?wsx version="1.0"?>
<smil>
   <media src="%requestedUrl%"/>
</smil>
```

Come potete vedere, si tratta di un file XML composto con un set ridotto del linguaggio SMIL. Questa prima versione del file wsx che vi presentiamo è molto semplice da interpretare. Dice semplicemente che ogni volta che un utente richiede un servizio appartenente a quel publishing point, deve essere inviato al client proprio il servizio richiesto.
Supponiamo ad esempio che il publishing points contenga il servizio *"Kenwood.wmw"*. Richiamando *mms://localhost/TG200406/kenwood.wmv* verrebbe visualizzato proprio il servizio in questione. Ma non sempre è questo il comportamento che ci attendiamo. Ad esempio vorremmo che qualunque fosse il servizio richiamato all'interno di un publishing points, venga sempre visualizzato prima di esso un filmato di sigla e uno di coda. Vediamo come cambiare il file in questione per realizzare quanto detto:

```
<?wsx version="1.0"?>
<smil>
   <media role="advertisement" noSkip="TRUE"
            src="C:\WMPub\WMRoot\TtimeTV.wmv"/>
   <media src="%requestedUrl%"/>
   <media role="advertisement" noSkip="TRUE"
            src="C:\WMPub\WMRoot\TtimeTV.wmv"/>
</smil>
```

Se provate adesso a connettervi all'indirizzo *mms://localhost/TG200406/kenwood.wmv* o a qualunque servizio appartenente al publishing point *TG200406* scoprirete che verrà sempre visualizzata una sigla, poi il servizio richiesto, poi la una sigla di coda. Potete ovviamente aggiungere quanti servizi volete alla directory e automaticamente saranno inclusi nel publishing points e visualizzati nel browser in ordine alfabetico. Se volete potete cambiarne l'ordine proprio attraverso la playlist.
Quello che è importante sottolineare in questa modifica fatta al file *wsx* è relativa all'introduzione della parola chiave advertisement seguita dall'argomento *noSkip* che realizza appunto l'effetto desiderato.
Volendo possiamo complicarci ancora la vita, ad esempio modificando la playlist nel seguente modo:

```
<?wsx version="1.0"?>
<smil>
   <media role="advertisement" noSkip="TRUE"
            src="C:\WMPub\WMRoot\TtimeTV.wmv"/>
   <media src="%requestedUrl%">
      <clientData title="Play" bannerAbstract=
                   "Playgeneration" bannerInfoURL=
         "http://www.edmaster.it/?job=prodotti&amp;
                            id=64" bannerURL=
   "http://www.edmaster.it/banner/468x60_play.gif"/>
   </media>
   <media role="advertisement" noSkip="TRUE"
            src="C:\WMPub\WMRoot\TtimeTV.wmv"/>
</smil>
```

Così facendo abbiamo comunicato al server di streaming che desideriamo che un banner pubblicitario sia proiettato in uno spazio appositamente riservato all'interno del player. Notate che abbiamo realizzato questo inserendo un nodo *clientData* all'interno di media. In realtà in una playlist composta da più media è possibile inserire il *clientData* come leaf di ciascun media, di modo che ogni filmato possa essere associato ad una campagna banner specifica. Infine è da segnalare la possibilità di creare una playlist dinamica. Dove ad esempio le sigle iniziali e finali variano sulla base di un AD Server che sponsorizza la trasmissione. Supponiamo ad esempio di avere 10 Sponsor per la trasmissione *X* e che ciascuno sponsor abbia richiesto che il proprio spot pubblicitario sia visualizzato *Y* volte. È ovvio che non possiamo mettere tutti gli sponsor nel file smil sotto la forma di advertising, perché ogni volta dovremmo mostrare 10 filmati prima di ottenere il servizio richiesto, inoltre è molto probabile che lo sponsor voglia ottenere un report delle visualizzazioni etc. Perciò sarebbe opportuno avere un server AD separato che scegliesse quale filmato mostrare prima del servizio richiesto. Proviamo a modificare il file *smil* come segue:

```
<?wsx version="1.0"?>
<smil>
   <media role="advertisement" noSkip="TRUE"
       src="httpd://localhost/DynamicPlaylist.aspx"/>
   <media src="%requestedUrl%">
      <clientData title="Play" bannerAbstract=
                   "Playgeneration" bannerInfoURL=
         "http://www.edmaster.it/?job=prodotti&amp;
                            id=64" bannerURL=
   "http://www.edmaster.it/banner/468x60_play.gif"/>
   </media>
   <media role="advertisement" noSkip="TRUE"
            src="C:\WMPub\WMRoot\TtimeTV.wmv"/>
</smil>
```

Abbiamo cambiato il src del primo advertising fornendogli un indirizzo httpd dinamico. Questo fa sì che il contenuto del file da visualizzare venga adesso prelevato dal file *DynamicPlaylist.aspx*. Questo file è un file asp.net che può contenere una qualsiasi logica per il reperimento del file *wmv* da visualizzare, pur-

ché al suo interno alla fine sia contenuta qualche istruzione del genere

```
Response.Write("<?wsx version=\"1.0\"?>\r\n");

Response.Write("<smil>\r\n");

Response.Write(" <media src=\"" + ads[index] + "\"
          noSkip=\"true\"" + "role=\"Advertisement\"
                                              />\r\n");

Response.Write("</smil>\r\n");
```

che appunto indica al server quale file di advertisment deve essere visualizzato

INSERIRE IL PLAYER IN UNA PAGINA HTML

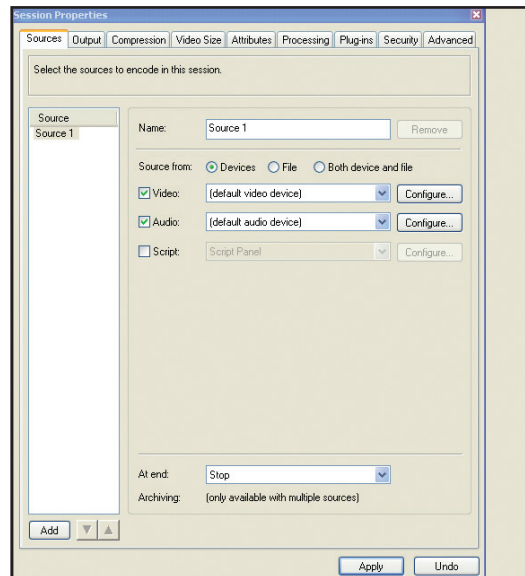
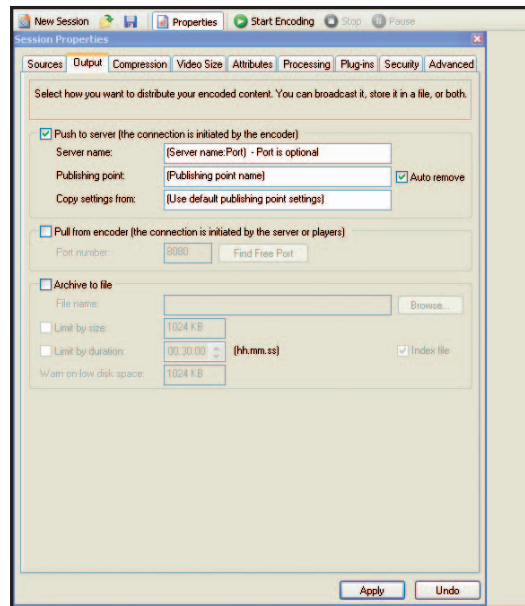
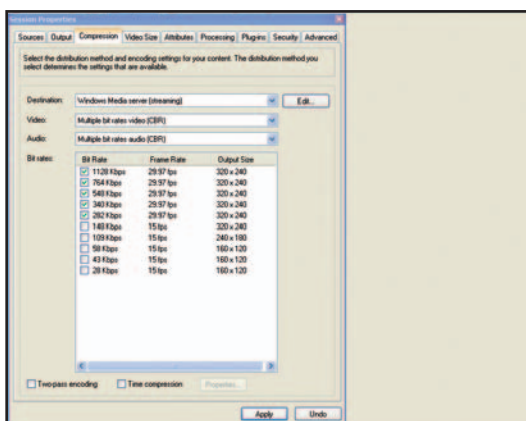
Anche in questo caso ci viene in aiuto il browser. Se date uno sguardo a quanto prodotto dal wizard vi accorgete che avete un file .html contenente uno scheletro che potete riutilizzare all'interno tipicamente di *inetpub/wwwroot*. Vi riproponiamo semplicemente qualche spezzone del codice tanto per farvi rendere conto di come questa tecnica viene applicata:

```
<script language="Javascript">
      if( g_bNetscape )
      {
          document.writeln(
            "<APPLET mayscript code=WMPNS.WMP
                            name=WMP1 width=300
                        height=200 MAYSCRIPT >" );
      }
      </script>
          <OBJECT CLASSID="clsid:6BF52A52-
                   394A-11D3-B153-00C04F79FAA6"
                                       ID="WMP">
         <PARAM NAME="Name" VALUE="WMP1">
         <PARAM NAME="URL" VALUE=
           "mms://JACO2003SERVER/TG200406">
      </OBJECT>
      </APPLET>
      </td>
```

LIVE STREAMING CON WINDOWS MEDIA ENCODER

Siamo finalmente giunti al momento in cui possiamo effettuare la prima trasmissione video in diretta televisiva. Lo scenario è quello di uno studio televisivo presente ad esempio a Roma che trasmette il video a un server in housing a Milano, al quale a loro volta i client Windows Media Player si connettono per poter visualizzare la trasmissione. Iniziamo lanciando il Windows Media Encoder e selezionando una custom session.

1 Portiamoci nella scheda di configurazione e in sources settiamo:

Name = test
Source from device (o from file a seconda di qual è il vostro dispositivo di ingresso)

2 nella *tabsheet output* selezioniamo *push to server*, in *server name* inseriamo il nome del server più una porta libera, infine il nome del *publishing point*, spuntiamo anche il check *autoremove*

3 spostiamoci nella tabsheet *compression*, scegliamo *CBR* per ottenere i vari formati di compressione e spuntiamo quelli che ci interessano

4 clicchamo su *apply* e diamo il via alla trasmissione. Se tutto è andato a buon fine un nuovo publishing point verrà creato automaticamente sul server e la trasmissione sarà visibile in broadcast. In sostanza è l'encoder che si connette al server e crea il publishing point, di modo che nessun intervento debba essere richiesto al sistemista.

Ovviamente il nostro scopo adesso è replicare la stessa situazione fornendo al nostro cliente non l'intero encoder ma un'interfaccia semplificata che gli consenta di gestire facilmente l'invio di uno stream al server.

IL WINDOWS MEDIA ENCODER SDK

Sul sito di Msdn all'indirizzo *http://www.microsoft.com/downloads/details.aspx?familyid=000a16f5-d62b-4303-bb22-f0c0861be25b&-displaylang=en* è disponibile il Windows Media Encoder sdk. Una volta installato ci mette a disposizione alcuni esempi interessanti per interagire con gli strumenti dell'encoder.
Ad esempio viene distribuita un'applicazione di scheduling che consente di schedulare l'encoding di file contenuti in una directory ad intervalli di tempo ben definiti. Oppure un'applicazione web per il remote control di un sistema di encoding. Ciò che è più importante ci mette a disposizione diversi oggetti COM da poter utilizzare nelle nostre applicazioni. Creare un'applicazione che invii un flusso in streaming a un Windows Media Server è abbastanza semplice.
Prima di tutto bisogna referenziare all'interno di un progetto .NET l'oggetto COM Windows Media Encoder, nell'ordine è necessario poi eseguire i seguenti passi:

1) Creare un oggetto *encoder*

2) Creare un gruppo che identifichi lo stream che si vuole mandare in trasmissione

3) Specificare il metodo di streaming

4) Avviare l'encoding

Un esempio di applicazione Visual Basic.NET che fa tutto questo è la seguente:

```
Imports WMEncoderLib
Public Class Form1
    Dim Encoder As WMEncoder
    Dim SrcGrpColl As IWMEncSourceGroupCollection
    Dim SrcGrp As IWMEncSourceGroup
    Dim SrcAud As IWMEncSource
    Dim SrcVid As IWMEncVideoSource
    Dim ProColl As IWMEncProfileCollection
    Dim Pro As IWMEncProfile
    Dim PushDist As IWMEncPushDistribution
    Dim strServerName As String
    Dim strPubPoint As String
    Dim strPubTemplate As String
    Dim MyNSCFile As String
    Dim MyNSCURL As String
    Dim MyASXFile As String

    Private Sub Form1_Load(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles MyBase.Load
            'Crea l'oggetto WMencoder
        Encoder = New WMEncoder
    End Sub

    Private Sub Button1_Click(ByVal sender As
```

E SE NON HO WINDOWS SERVER 2003?

La trasmissione di video su Internet sta diventando abbastanza comune da avere favorito la proliferazione di provider di Hosting che mettono a disposizione Windows Server 2003 per i propri clienti. In sostanza mentre il Windows Media Encoder continuerà a risiedere sulla vostra macchina, potrete connettervi in remoto a un Windows Media Services per effettuare la trasmissione in Broadcasting o in Unicasting.
In tal caso una normale connessione ADSL è tipicamente sufficiente per trasmettere il video in modo fluido. Tuttavia è importante che la disponibilità di banda del vostro provider di Hosting sia sufficiente a coprire in modo ottimale la ritrasmissione dei dati per molti clienti in contemporanea. Quando vi affidate a un hoster tipicamente usufruite di un servizio "Shared" quindi non siete solo voi a trasmettere. Se non volete che la connessione proceda a scatti dovete assicurarvi con il vostro provider che sia in grado di sostenere questo compito.

```
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
If OpenFileDialog1.ShowDialog() =
Windows.Forms.DialogResult.OK Then
ListBox1.Items.Add(
OpenFileDialog1.FileName)
End If

End Sub

Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
SrcGrpColl = Encoder.SourceGroupCollection

' Crea un gruppo chiamato SG_1.
SrcGrp = SrcGrpColl.Add("SG_1")

' Specifica la sorgente
'Qui lasciasmo commentata la linea che recupera
anche una sorgente audio
'SrcAud = SrcGrp.AddSource(
WMENC_SOURCE_TYPEWMENC_AUDIO)
SrcVid = SrcGrp.AddSource(
WMENC_SOURCE_TYPE.WMENC_VIDEO)

' Specifica il path del file sorgente
' nel nostro caso lo recuperiamo dal
contenuto di una listbox
'SrcAud.SetInput("c:\audio.wav")
SrcVid.SetInput(ListBox1.Items(0).ToString())

'Se vogliamo prendere come sorgente un device
'SrcAud.SetInput("device://default_audio_device")
'SrcVid.SetInput("device://default_video_device")

' Stabilice il profilo da utilizzare per la codifica.
ProColl = Encoder.ProfileCollection
Pro = ProColl.Item(2)
SrcGrp.Profile = Pro

' Definisce l'encoding in  broadcast.
PushDist = Encoder.Broadcast

' Definisce il nome del publishing Point e la porta
strServerName = "localhost:8888"
strPubPoint = "publicationpoint"

' Rimuove il publishing point quando la
' Trasmissione è finita
PushDist.AutoRemovePublishingPoint = True

PushDist.ServerName = strServerName
PushDist.PublishingPoint = strPubPoint
Encoder.PrepareToEncode(True)
' Start encoding.
Encoder.Start()
MsgBox("Click OK to stop broadcasting.")

End Sub
End Class
```

Come vedete, replichiamo in modo programmatico gli stessi indentici passaggi che abbiamo fatto con l'encoder. Il nostro scheletro di applicazione è pronto! Potete modificarlo come meglio credete per offrire ai vostri clienti il miglior servizio possibile.

CONCLUSIONI

Il video over IP è probabilmente il mezzo più trascurato rispetto al digitale terrestre e al satellitare. Probabilmente perché ancora manca quell'immediatezza d'uso che invece la televisione tradizionale ha. Tuttavia al crescere della disponibilità della banda e al migliorare dell'integrazione della tv tradizionale con il protocollo IP siamo certi che questo mezzo di trasmissione sarà un grande competitor per tutti gli altri. Naturalmente al momento esistono ancora alcune limitazioni che non consentono un uso fruibile del mezzo da parte delle famiglie, come invece accade per la TV tradizionale, prima fra tutte la disponibilità della banda. Tuttavia al crescere dell'infrastruttura di rete sarà inevitabile considerare questo tipo di trasmissione dati come una fra le candidate ad essere il mezzo televisivo del futuro. Noi programmator dovremo farci trovare pronti a supportare i clienti che vorranno tuffarsi in questo campo.

USO DEI PROTOCOLLI

Può capitare in qualche caso che la stazione trasmittente o ricevente sia costretta a lavorare all'interno di un network protetto da firewall. Ci vengono in aiuto i vari protocolli supportati da Windows Media Services. In particolare WMS usa due protocolli per trasmettere in unicast i dati ai client.

RTSP: acronimo di Real Time Streaming Protocol
MMS: acronimo di Microsoft Media Server Protocol
RTSP è un protocollo studiato appositamente per consentire la trasmissione dei dati in Real Time, MMS viceversa è un protocollo proprietario studiato che agevola le connessioni con client di tipo Windows Media Player.
Infine è possibile effettuare lo streaming anche su protocollo http tipicamente usato dai servizi web e disponibile solitamente anche quando il proprio network è coperto da firewall.
Il vantaggio di poter usufruire di protocolli come RTSP o MMS sta soprattutto nel fatto di poter utilizzare quella che viene detta la classica funzione di "Rollover" , che consente un'ottimizzazione della trasmissione tra client e server e in casi di fault è in grado di rinegoziare in modo ottimale la ritrasmissione dei contenuti

PHP ALLA RICERCA DELLE PRESTAZIONI!

IN QUESTO ARTICOLO, IMPAREREMO ALCUNE COSE SUI MECCANISMI INTERNI DI FUNZIONAMENTO DI PHP E UTILIZZEREMO UN'ESTENSIONE CHE RADDOPPIA LA VELOCITÀ DI ESECUZIONE DEGLI SCRIPT! PRONTI, PARTENZA... VIA

Uno dei vantaggi più grandi di PHP sta nell'essere un linguaggio di scripting. Cosa vuol dire? Molto semplicemente che potete scrivere le vostre applicazioni in un notepad salvandole in un classico file di testo, potete piazzare i vostri script su un server, e magicamente tutto funzionerà senza dover compilare, aggiungere DLL o altro. Ma, esattamente, come funziona questo meccanismo di scripting? Se ci pensate la soluzione è abbastanza semplice. Sul nostro web server abbiamo un file di testo, ad esempio *index.php*, che potrebbe contenere qualcosa del genere:

```
<?
phpinfo();
?>
```

quando un utente punta il browser all'indirizzo che espone questo file, ad esempio *http://localhost /~apc/index.php*, il server riconosce che è stato richiamato un file con estensione PHP e richiama lo "zend engine".
L'engine effettua un parsing del file in questione, lo compila e genera un opcode, ovvero il corrispondente in linguaggio macchina di quanto espresso dallo script con il linguaggio ad alto livello. A questo punto, chiaramente, la macchina sarà in grado di eseguire il codice. Quando lo script termina l'opcode viene distrutto! Cosa vuol dire tutto questo? molto semplicemente che ogni volta che uno script viene richiamato deve riavviarsi il processo di compilazione, questo ovviamente consuma numerosi cicli macchina con conseguente perdita di tempo. E se in uno script ci fossero file inclusi, cosa succederebbe? Considerate ad esempio il seguente spezzone di codice:

```
<?
        require_once('db.php');
        require_once('extension.php');
        print "test eseguito";
?>
```

in questo caso:

1) verrebbe compilato lo script principale e mandato in esecuzione;
2) durante l'esecuzione si incontra la prima direttiva di inclusione, quindi anche il secondo file viene compilato ed eseguito con la stessa logica;
3) si incontra poi la seconda direttiva di inclusione, anche in questo caso è necessario ricompilare;
4) infine lo script viene eseguito e completato correttamente.

Tutto questo ogni volta che lo script viene richiamato. Immaginate per centinaia di accessi concorrenti quanti cicli macchina devono essere impiegati per soddisfare queste esigenze.

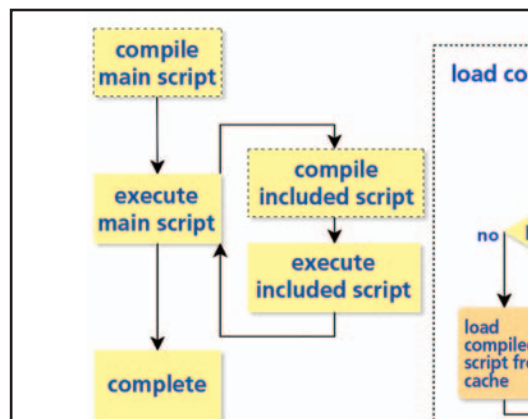



Fig. 1: ***Diagramma di flusso dell'esecuzione classica di uno script PHP***

LA SOLUZIONE APC

APC ovvero *Alternative PHP Cache* è un'estensione di PHP che consente di "salvare" uno script compilato in un'area di memoria e di richiamarlo ogni volta che ce ne fosse bisogno senza doverlo ricompilare. Con APC il diagramma di flusso dell'esempio precedente diventerebbe

1) viene "caricato" il file *index.php;*
2) si controlla se questo file compilato è già pre-

sente nella cache;

3) se il file non è stato compilato, lo si compila e lo si mette nella cache;
4) se il file è stato compilato si controlla se è stato modificato dopo l'ultima compilazione;
5) se è stato modificato lo si ricompila e lo si salva di nuovo nella cache;
6) se non è stato modificato lo si carica dalla cache e lo si esegue.

questa procedura viene eseguita ricorsivamente per ogni file incluso nello script, con evidenti guadagni in termini di tempo

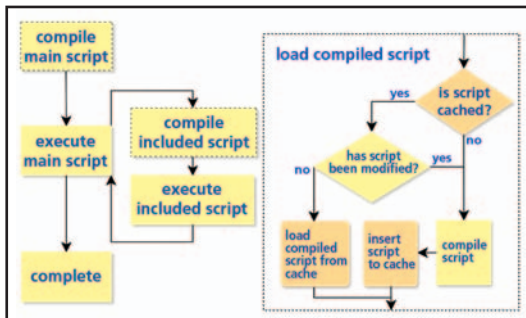



Fig. 2: ***Il diagramma di funzionamento dell'esecuzione di uno script con l'estensione APC installata***

DALLA TEORIA ALLA PRATICA

APC è un'estensione di PHP. Si mormora che sarà inserita di default nel nascituro PHP 6, fino ad oggi siamo ancora costretti ad installarla dalla PECL. Per gli utenti Linux questo si riduce ad eseguire dalla bash il seguente comando:

```
pecl install apc
```

per gli utenti Windows sarà necessario invece scaricare la apc.dll dal repository *http://it.php.net/get/pecl-5.1.2-Win32.zip/from/a/mirror*, in tutti e due i casi sarà necessario aggiungere l'estensione nel php.ini, come segue:

```
[per gli utenti windows]
extension=apc.dll
```

```
[per gli utenti linux]
extension=apc.so
```

e naturalmente riavviare Apache o IIS se si sta usando il Web Server di Microsoft. Nessun'altra configurazione è necessaria per attivare APC.

Per essere sicuri che tutto funzioni, creiamo un file index.php con all'interno le seguenti istruzioni:

```
<?
```

```
phpinfo();
?>
```

puntiamo il browser all'indirizzo che espone il file in questione e se tutto è andato a buon fine, troveremo APC tra le estensioni attive, come mostrato in **Figura 3**.

apc

APC Support	enabled
Version	3.0.10
MMAP Support	Enabled
MMAP File Mask	no value
Revision	$Revision: 3.84 $
Build Date	Apr 6 2006 09:08:37

Directive	Local Value	Master Value
apc.cache_by_default	On	On
apc.enable_cli	Off	Off
apc.enabled	On	On
apc.file_update_protection	2	2
apc.filters	no value	no value
apc.gc_ttl	3600	3600
apc.max_file_size	1M	1M
apc.mmap_file_mask	no value	no value
apc.num_files_hint	1000	1000
apc.optimization	Off	Off
apc.shm_segments	1	1
apc.shm_size	30	30
apc.slam_defense	Off	Off
apc.stat	On	On
apc.ttl	0	0

Fig. 3: ***L'estensione APC è installata e funzionante***

CODE PROFILING

Per poter testare la validità della soluzione APC avremo bisogno di un qualche software che ci dia una valutazione numerica precisa di quale sia il guadagno nell'utilizzo di questa tecnica. Per i nostri scopi utilizzeremo *xdebug*. Neanche a farlo apposta si tratta di un'ulteriore estensione di PHP che ci restituisce fra gli altri il tempo utilizzato per la generazione dell'opcode. L'installazione di *Xdebug* segue la stessa procedura che abbiamo utilizzato per APC. Per gli utenti Linux:

```
pecl install xdebug
```

per gli utenti Windows è necessario scaricare la dll dall'indirizzo *http://pecl4win.php.net/ext.php/php_xdebug.dll*.

Anche in questo caso è necessario attivare l'estensione nel *php.ini* e riavviare il web server. Uno script che potete utilizzare per notare la differenza nella generazione dell'opcode è il seguente:

```
<?php
xdebug_start_profiling();

for($i = 0; $i < 10000; $i++) {
 echo 'PHP for life!';
}

xdebug_dump_function_profile();
xdebug_stop_profiling();
?>
```

e controllare in output il valore contenuto in opco-

NOTA

POTENZA DI XDEBUG

In questo articolo abbiamo utilizzato solo alcune delle caratteristiche di questo imponente debugger. Con Xdebug si possono ottenere informazioni dettagliate sullo stato della memoria, sulla velocità dell'esecuzione, sull'occupazione dello stack e una serie di altre informazioni incredibilmente utili in fase di fine tuning dell'applicazione.
Per Xdebug il sito di riferimento è:
http://www.xdebug.org/

de. Naturalmente si tratta di uno script abbastanza semplice, per cui i valori saranno molto piccoli, ma può dare già la dimensione di quanto APC possa migliorare le prestazioni dei nostri script. Per ottenere una valutazione migliore potete provare con script più complessi.

DEBUGGING DI APC

Cerchiamo di andare un po' più in profondità per capire cosa esattamente succede quando viene richiamato uno script e l'estensione APC è attivata. Per l'occasione creeremo due script, il primo si chiamerà control.php e conterrà il seguente codice:

```
<?
print "<pre>";
print_r(apc_cache_info());
print "</pre>";
?>
```

il secondo si chiamerà *test.php* e conterrà il seguente codice:

```
<?php

xdebug_start_profiling();
require_once('utils.php');
try {
    $msg = new utils();
    $msg->messaggio="messaggio di prova";
} catch (Exception $e) {
    print $e->getMessage();
    exit();
}
for ($i = 0; $i < 5000; $i++) {
  echo str_repeat($msg->messaggio, rand(1, 3));
}

xdebug_dump_function_profile();
xdebug_stop_profiling();
?>
```

avremo ovviamente bisogno di un terzo file chiamo *utils.php* in cui defineremo qualche classe d'appoggio, in particolare:

```
<?
class utils {
            private $messaggio;
            private function __get($property) {
      if($property == "messaggio") {
          return $this->$property;
      }
      else {
          throw new Exception("No such property:
                                                $property");
        }
    }

    private function __set($property,$value) {
        if($property == "messaggio") {
            $this->$property=$value;
        }
        else {
            throw new Exception("No such property:
                                                $property");
        }
    }
}
?>
```

si tratta anche questo di uno script sufficientemente banale che non fa altro che ripetere a video un messaggio. Lo abbiamo semplicemente complicato un poco creando una classe di supporto, instanziando qualche oggetto e aggiungendo una funzione matematica, semplicemente per innalzare la complessità dello script ed ottenere qualche risultato più utile per i nostri fini. Eseguiamo lo script test *.php* un paio di volte e poi lo script *control.php* che è quello che ci interessa in questo momento. La funzione *apc_cache_info()* ci restituisce un array contenente le informazione sugli opcdode attualmente disponibili nella cache:

```
[cache_list] =>
  Array (
    [0] => Array (
      [filename] => /home/fabio/public_html
                                    /apc/test.php
      [device] => 770
      [inode] => 324743
      [type] => file
      [num_hits] => 1
      [mtime] => 1144319900
      [creation_time] => 1144320273
      [deletion_time] => 0
      [access_time] => 1144320279
      [ref_count] => 0
      [mem_size] => 5298 )
    [1] => Array (
      [filename] => /home/fabio/public_html
                                    /apc/utils.php
      [device] => 770
      [inode] => 347769
      [type] => file
      [num_hits] => 1
      [mtime] => 1144319636
      [creation_time] => 1144320273
      [deletion_time] => 0
      [access_time] => 1144320279
      [ref_count] => 0
      [mem_size] => 6442 )
  [2] => Array (
```

```
      [filename] => /home/fabio/public_html
                              /apc/controllo.php
        [device] => 770
        [inode] => 385758
        [type] => file
        [num_hits] => 0
        [mtime] => 1144316969
        [creation_time] => 1144320299
        [deletion_time] => 0
        [access_time] => 1144320299
        [ref_count] => 0 [mem_size] => 1616 )
```

Notate come nella cache vengono inseriti anche gli script compilati *utils.php* e *controllo.php*, ovvero rispettivamente lo script incluso e l'ultimo script eseguito.

GESTIONE DELLA CACHE

Una volta stabilite le modalità per l'installazione e per il testing di APC, l'80% del lavoro è fatto. Lo scopo di APC è innalzare le prestazioni utilizzando una cache per l'opcode, come già detto più volte.
È anche vero che questa estensione espone alcuni metodi e proprietà per la gestione della cache che possono risultare interessanti. Consideriamo il seguente spezzone di codice:

```
<?
define('UNO',1);
define('DUE',2);
define('TRE',3);
echo UNO, DUE, TRE;
?>
```

è noto che l'istruzione di define delle costanti è parecchio lenta. Se volessimo sfruttare a fondo i meccanismi messi a disposizione di APC potremmo riscrivere il codice precedente come segue:

```
<?
$constants = array(
    'UNO_C'  => 1,
    'DUE_C'  => 2,
    'TRE_C' => 3,
 );
 apc_define_constants('numbers', $constants);
 apc_load_constants('numbers');

 echo UNO_C, DUE_C, TRE_C;
?>
```

un altro esempio interessante d'utilizzo di APC riguarda le variabili. Consideriamo il seguente script:

```
<?
$bar = 'BAR';
apc_store('stringa di prova', $bar);
var_dump(apc_fetch('foo'));
?>
```

come si vede, semplicemente definisce una variabile *$bar* e poi conserva il suo contenuto in un'area globale della cache di APC attribuendogli un'etichetta chiamata *'foo'*. Il risultato di questa operazione è che adesso il contenuto di *foo*, non solo sarà conservato nella cache, ma addirittura potrà essere recuperato da altri script al di fuori di quello corrente e della stessa sessione. Il che significa che alla prima esecuzione dello script da parte di qualunque utente, verrà valorizzata la variabile in questione e che tutti gli altri utenti potranno recuperare il suo contenuto direttamente dalla cache senza dover compiere nuove operazioni di calcolo.
Ovviamente questa caratteristica deve essere utilizzata con una certa attenzione se non si vuole incorrere in problemi di sicurezza, ma è altrettanto vero che se utilizzata nel giusto contesto, velocizza incredibilmente le prestazioni di PHP. Allo stesso modo la variabile può essere rimossa dalla cache tramite la funzione *apc_delete($key)*.

CONCLUSIONI

APC è uno strumento realmente semplice da utilizzare, ma molto potente. In un ambiente di test, dove gli accessi contemporanei, sono in un numero necessariamente basso non noterete il profondo cambiamento di prestazioni, ma in ambienti di produzione laddove i siti generano un numero alto di accessi contemporanei vi accorgerete di quanto adottare meccanismi di questo genere possono aumentare le prestazioni. Certamente, programmare utilizzando una cache per l'opcode comporta una dose maggiore di attenzione alle politiche di sicurezza, tuttavia adottando le dovute accortezze sicuramente ne ricaverete un beneficio, specialmente nei sistemi moderni dove decisamente non ci sono più problemi di limitata disponibilità della memoria e si ricercano invece le massime prestazioni in termini di risposta da parte delle web application.

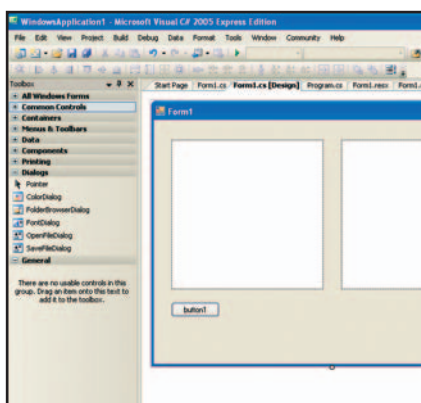
PROGRAMMAZIONE AD OGGETTI

In questo articolo non sarà passato inosservato che in uno degli esempi abbiamo utilizzato un elegante metodo per la creazione di setter e getter universali. Chi è abituato a programmare ad oggetti, sa che è utile mantenere le variabili locali alla classe come private, per poi utilizzare dei setter e dei getter per attribuire o recuperare un valore dalle variabili in questione.
È proprio il metodo che abbiamo usato noi, in più abbiamo proposto una soluzione sempre valida, qualunque siano le proprietà da gestire all'interno di una classe.

IO PROGRAMMO BY EXAMPLE

IMPARA A PROGRAMMARE IN MODO PRATICO E DIVERTENTE, CON GLI ESEMPI PASSO PASSO CHE TI GUIDANO ALLA COSTRUZIONE DEL CODICE

COME POSSO RIDIMENSIONARE UN'IMMAGINE?

.Net mette a disposizione dei metodi abbastanza semplici, tuttavia se non si vuole perdere in qualità è necessario adottare qualche accorgimento...

C# / VISUAL BASIC

COME POSSO ESTRARRE IL CONTENUTO DI UN FILE ZIP?

Utilizzeremo le sharpziplib, libreri opensource che gestiscono in modo efficace e completo file compressi con un qualunque tipo di formato

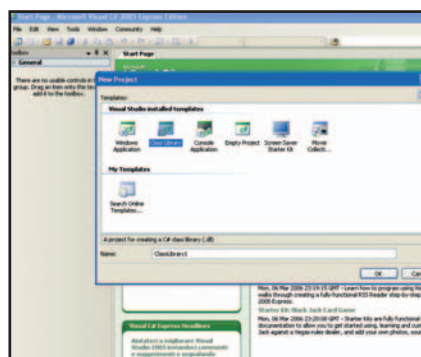
PHP&NUSOAP / .NET

COME POSSO USARE ACCES COME CLIENT DI SQL SERVER 2005?

Può essere una soluzione molto comoda per creare rapidamente delle maschere o gestire un database in modalità struttura o inserimento dati. La procedura è semplicissima

C# / VISUAL BASIC.NET

COME POSSO SCRIVERE UNA STORED PROCEDURE IN MANAGED CODE?

A partire da visual studio 2005 è possibile scrivere una ST direttamente dal CLR, questo consente di poter utilizzare un unico ambiente per tutte le operazioni

VUOI INVIARE UN ESEMPIO?

Se sei un programmatore esperto ed hai risolto un problema, puoi aiutare gli altri pubblicando il tuo codice. Proponi i tuoi esempi scrivendo a ioprogrammo@edmaster.it

Come contattarci?

Alla nostra redazione arriva spesso un considerevole numero di email riguardante temi specifici. Per consentirci di rispondere velocemente e in modo adeguato alle vostre domande abbiamo elaborato una FAQ – Frequently Ask Question – o risposte alle domande frequenti in italiano, di modo che possiate indirizzare le vostre richieste in modo mirato.

Problemi sugli abbonamenti

Se la tua domanda ha a che fare con una delle seguenti:

- Vorrei abbonarmi alla rivista, che devo fare?
- Sono un abbonato e non ho ricevuto la rivista, a chi devo rivolgermi?
- Sono abbonato ma la posta non mi consegna regolarmente la rivista, a chi devo rivolgermi?

Contatta **abbonamenti@edmaster.it** specificando che sei interessato a ioProgrammo. Lascia il tuo indirizzo email e indica il numero dal quale vorresti far partire l'abbonamento. Verrai contattato al più presto. Oppure puoi chiamare lo **02 831212**

Problemi sugli allegati

Se riscontri un problema del tipo:

- Ho acquistato ioProgrammo ed il Cdrom al suo interno non funziona. Chi me lo sostituisce?
- Ho acquistato ioProgrammo ma non ho trovato il cd/dvd all'interno, come posso ottenerlo?
- Vorrei avere alcuni arretrati di ioProgrammo come faccio?

Contatta **servizioclienti@edmaster.it**
Non dimenticare di specificare il numero di copertina di ioProgrammo e la versione: con libro o senza libro. Oppure telefona allo **02 831212**

Assistenza tecnica

Se il tuo problema è un problema di programmazione del tipo:

- Come faccio a mandare una mail da PHP?
- Come si instanzia una variabile in c++?
- Come faccio a creare una pagina ASP.NET

o un qualunque altro tipo di problema relativo a tecniche di programmazione, esplicita la tua domanda sul nostro forum: **http://forum.ioprogrammo.it**, uno dei nostri esperti ti risponderà. Le domande più interessanti saranno anche pubblicate in questa rubrica.

Problemi sul codice all'interno del CD

Se la tua domanda è la seguente

- Non ho trovato il codice relativo all'articolo all'interno del cd

Consulta la nostra sezione download all'indirizzo **http://cdrom.ioprogrammo.it**, nei rari casi in cui il codice collegato ad un articolo non sia presente nel cdrom, senza dubbio verrà reso disponibile sul nostro sito.

Idee e suggerimenti

Se sei un programmatore esperto e vuoi proporti come articolista per ioProgrammo, oppure se hai suggerimenti su come migliorare la rivista, se vuoi inviarci un trucco suggerendolo per la rubrica tips & tricks invia una email a
ioprogrammo@edmaster.it

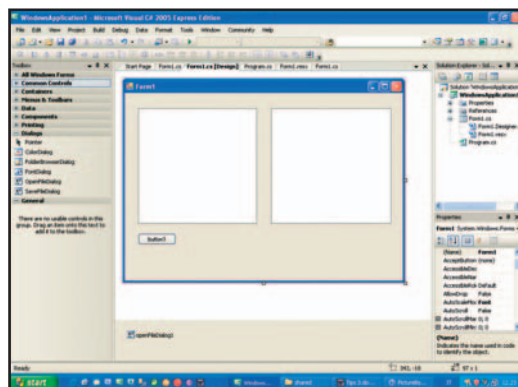
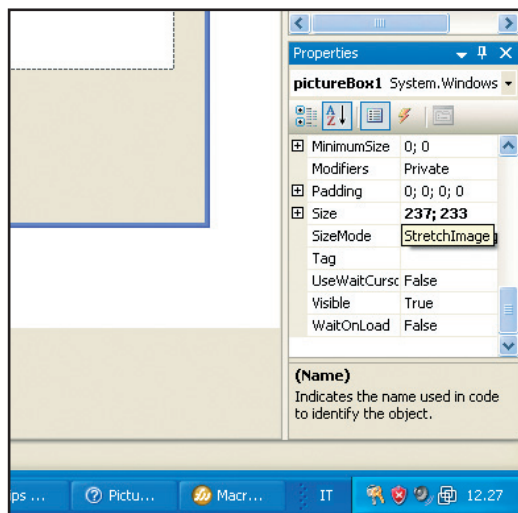
COME POSSO RIDIMENSIONARE UN'IMMAGINE?

.NET METTE A DISPOSIZIONE DEI METODI ABBASTANZA SEMPLICI, TUTTAVIA SE NON SI VUOLE PERDERE IN QUALITÀ È NECESSARIO ADOTTARE QUALCHE ACCORGIMENTO…

C#

VISUAL BASIC.NET

FACCIAMOLO IN C#

1 Creiamo una form composta da due PictureBox e da un bottone, trasciniamo inoltre sulla stessa form un componente openFileDialog dalla toolbox

2 Preoccupiamoci di settare la property *SizeMode* dell'immagine di sinistra a *"StrechImage"*

3 Clicchiamo due volte sul bottone per ottenere lo scheletro di gestione dell'evento *onClick*. Il codice da inserire sarà il seguente

```
private void button1_Click(object sender, EventArgs e)
{
  DialogResult result = openFileDialog1.ShowDialog();
  if (result == DialogResult.OK)
  {
    file = openFileDialog1.FileName;
    pictureBox1.Image = Image.FromFile(
                        openFileDialog1.FileName);
```

```
    resize();
  }
}
```

4 poco più in basso scriviamo il nostro metodo personalizzato per il resize

```
private void resize(){
        this.Cursor = Cursors.WaitCursor;
        Image oImg = Image.FromFile(file);
        Image oRes = new Bitmap(
            pictureBox2.Width, pictureBox2.Height);

        Graphics oGraphic =
                        Graphics.FromImage(oRes);

        oGraphic.CompositingQuality =
                        System.Drawing.Drawing2D
                    .CompositingQuality.HighQuality;

        oGraphic.SmoothingMode =
                        System.Drawing.Drawing2D
                        .SmoothingMode.HighQuality;

        oGraphic.InterpolationMode =
                        System.Drawing.Drawing2D
                .InterpolationMode.HighQualityBicubic;

        double a = (double)pictureBox2.Width
                            / (double)oImg.Width;
        double b = (double)pictureBox2.Height
                            / (double)oImg.Height;
        double scala = Math.Min(a, b);
        int dx = (int)(oImg.Width * scala), dy =
                            (int)(oImg.Height * scala);
        Rectangle oRectangle = new
            Rectangle((pictureBox2.Width - dx) / 2,
             (pictureBox2.Height - dy) / 2, dx, dy);
        Rectangle sRectangle = new Rectangle(
                    0, 0, oImg.Width, oImg.Height);

        oGraphic.DrawImage(oImg, oRectangle,
                    sRectangle, System.Drawing
                            .GraphicsUnit.Pixel);

        oGraphic.Dispose();
        if (pictureBox2.Image != null)
                {pictureBox2.Image.Dispose();
                  pictureBox2.Image = null; }
        pictureBox2.Image = oRes;
        this.Cursor = Cursors.Default;
```

```
    }
}
```

COME FUNZIONA?

Nel pannello di sinistra viene caricata l'immagine con i metodi consueti. Nel pannello di destra, per confronto carichiamo invece l'immagine con un resize più accurato. Inizialmente alla pressione del bottone appare una dialog box per la scelta del file di immagine da visualizzare. Il pannello di destra viene riempito come di consueto tramite il metodo Image.FromFile…, per riempire il pannello di destra invochiamo invece il nostro metodo resize(). In questo metodo viene creato un oggetto di classe Image riempito con il contenuto dell'immagine presa dall'hard disk. Viene poi creato un secondo oggetto di tipo Image chiamato oRes. Si crea un oggetto Graphics e si settano tutti i parametri per ottenere una conversione senza perdita di qualità. Quando tutto è settato correttamente, si assegna l'oggetto oRes alla seconda pictureBox. Il legame fra l'immagine originale e l'immagine ridimensionata è fornito dalle due linee

```
Graphics oGraphic = Graphics.FromImage(oRes);
oGraphic.DrawImage(oImg, oRectangle, sRectangle,
                    System.Drawing.GraphicsUnit.Pixel);
```

FACCIAMOLO IN VISUAL BASIC

Ripetere i passi uno e due come in C#, il codice di gestione del click sul bottone invece diventa

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                              Handles Button1.Click
        Dim result As DialogResult
        result = OpenFileDialog1.ShowDialog()
        If result = Windows.Forms.DialogResult.OK Then
            file = OpenFileDialog1.FileName
            PictureBox1.Image = Image.FromFile(
                          OpenFileDialog1.FileName)
            myresize()
        End If
    End Sub
```

VARIABILI PRIVATE

Sia in C# che in VB.net si fa uso di una variabile privata file, la si può dichiarare nella prima parte del codice come segue:

```
Private file As String
```

Nel caso di Visual Basic e

```
private string file;
```

nel caso di C#.

GLI IMPORT DA UTILIZZARE

Nel caso di C#

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Text;
using System.Windows.Forms;

Il metodo *myresize()* può essere riscritto nel seguente modo

```
Private Sub myresize()
        Me.Cursor = Cursors.WaitCursor
        Dim oImg As Image = Image.FromFile(file)
        Dim oRes As Image = New Bitmap(
                PictureBox2.Width, PictureBox2.Height)
        Dim oGraphic As Graphics =
                        Graphics.FromImage(oRes)
        oGraphic.CompositingQuality =
                        System.Drawing.Drawing2D
                    .CompositingQuality.HighQuality
        oGraphic.SmoothingMode =
                        System.Drawing.Drawing2D
                        .SmoothingMode.HighQuality
        oGraphic.InterpolationMode =
                        System.Drawing.Drawing2D
              .InterpolationMode.HighQualityBicubic
        Dim a As Double = (CDbl(PictureBox2.Width)
                            / CDbl(oImg.Width))
        Dim b As Double = (CDbl(PictureBox2.Height)
                            / CDbl(oImg.Height))
        Dim scala As Double = Math.Min(a, b)
        Dim dx As Integer = oImg.Width * scala
        Dim dy As Integer = oImg.Height * scala
        Dim oRectangle As Rectangle = New
              Rectangle((PictureBox2.Width - dx) / 2,
               (PictureBox2.Height - dy) / 2, dx, dy)
        Dim sRectangle As Rectangle = New
            Rectangle(0, 0, oImg.Width, oImg.Height)
        oGraphic.DrawImage(oImg, oRectangle,
                      sRectangle, System.Drawing
                                .GraphicsUnit.Pixel)
        oGraphic.Dispose()
        If Not (PictureBox2.Image Is Nothing) Then
            PictureBox2.Image.Dispose()
            PictureBox2.Image = Nothing

        End If

        PictureBox2.Image = oRes
        Me.Cursor = Cursors.Default

    End Sub
```

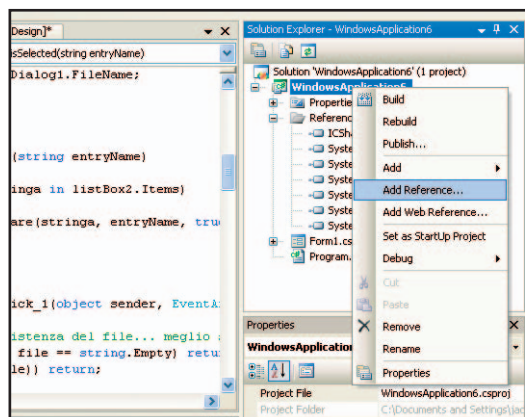
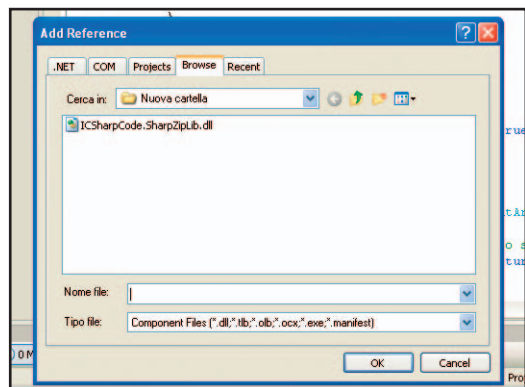
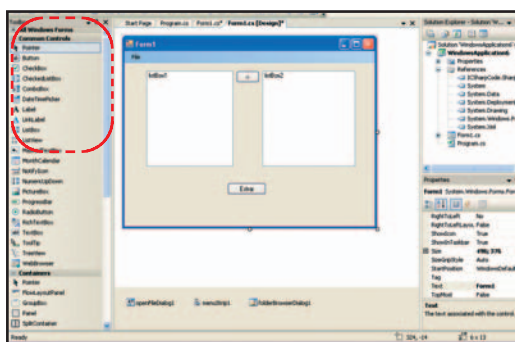
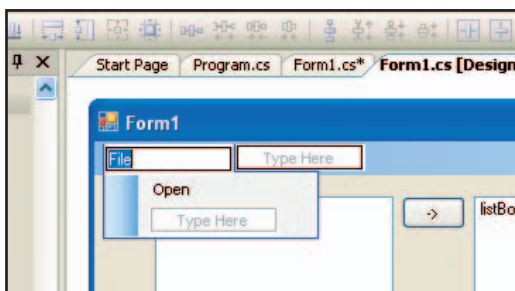
COME POSSO ESTRARRE IL CONTENUTO DI UN FILE ZIP?

UTILIZZEREMO LE SHARPZIPLIB, LIBRERIE OPENSOURCE CHE GESTISCONO IN MODO EFFICACE E COMPLETO FILE COMPRESSI CON UN QUALUNQUE TIPO DI FORMATO

DA DOVE SI SCARICANO LE SHARPZIPLIB?

L'Url di riferimento è http://www.icsharpcode.net/OpenSource/SharpZipLib/Download.aspx, da cui si possono scaricare le librerie già compilate, oppure il loro codice sorgente, sempre utile se si vogliono approfondire le tecniche di compressione. Il progetto è sviluppato utilizzando C#.

FACCIAMOLO IN C#

1 Dopo aver scaricato le *SharpZipLib* nella loro forma binaria, decomprimo lo zip e aggiungiamo il riferimento alla DLL nel progetto Visual Studio. Per farlo utilizziamo il tasto destro del mouse sul *soluzion explorer,* e clicchiamo alla voce *"Add Reference"*

2 Nella finestra che appare selezioniamo la tabsheet *"browse"* e sfogliamo l'hard disk alla ricerca del file *ICSharpCode.SharpZipLib.dll,* quando lo abbiamo trovato selezioniamolo e clicchiamo su *"Ok"*

3 Componiamo la form trascinando dalla toolbar, un menu, due listbox, due pulsanti, un *openFileDialog,* e un *folderBrowserDialog*

4 Organizziamo il menu di modo che sia composto da una voce *"File"* e un item *"Open"*

5 Dichiariamo una variabile privata *"file"* che punterà al nome del file zip da decomprimere

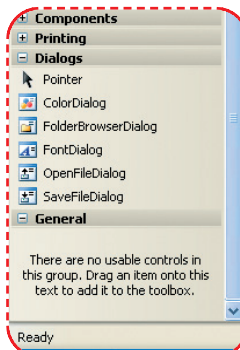
```
private string file;
```

6 Clicchiamo due volte sull'item *Open,* per generare lo scheletro di gestione dell'evento. Il nostro codice sarà il seguente

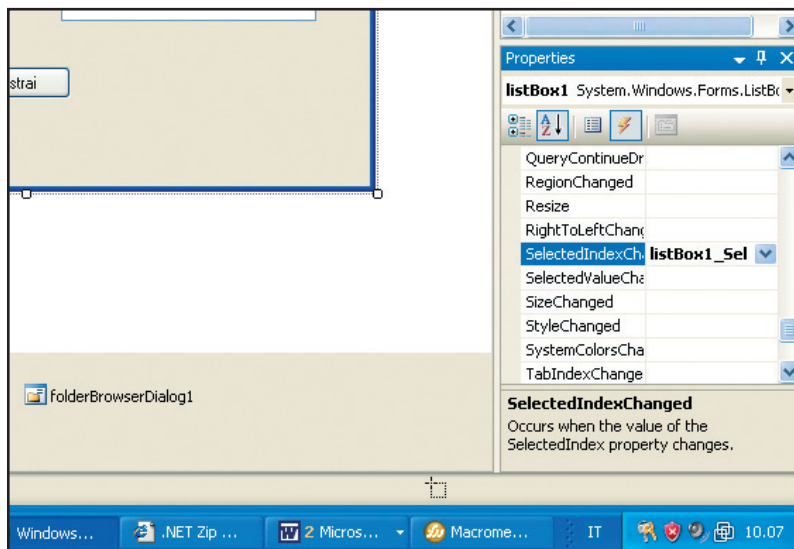
```
private void openToolStripMenuItem_Click(
                                object sender, EventArgs e)
        {
            DialogResult result =
                                openFileDialog1.ShowDialog();
            if (result == DialogResult.OK)
            {
                file = openFileDialog1.FileName;
                zipinfo();
            }
        }
```

7 Poco più in basso scriviamo il metodo *zipinfo* che provvederà a riempire la listbox1 con le

C#

VISUAL BASIC

informazioni recuperate dal file compresso selezionato al passo precedente

```
private void zipinfo() {
        listBox1.Items.Clear();
        listBox2.Items.Clear();
         button2.Enabled = false; button1.Enabled
                                          = false;
        ZipInputStream s = new
                ZipInputStream(File.OpenRead(file));
        ZipEntry theEntry;
        string pathproject=string.Empty;
        string pathimg=string.Empty;
        while ((theEntry = s.GetNextEntry()) != null)
        {
            listBox1.Items.Add(theEntry.Name);
        }
        s.Close(); s=null;
        theEntry=null;
    }
```

8 Selezioniamo la listbox1, e clicchiamo due volte sull'evento SelectedIndexChanged per generarne lo scheletro di gestione del codice. Il nostro codice sarà il seguente

```
private void listBox1_SelectedIndexChanged(
                          object sender, EventArgs e)
    {
        button2.Enabled = true;
    }
```

Clicchiamo due volte sul bottone che ci servirà per selezionare i file che vogliamo estrarre dallo zip selezionato, e anche di questo evento gestiamone lo stato con il seguente codice:

```
private void button2_Click(object sender, EventArgs e)
    {
        if (listBox1.SelectedIndex < 0) return;
        listBox2.Items.Add(listBox1.SelectedItem);
        listBox1.Items.Remove(
                              listBox1.SelectedItem);
        button2.Enabled = false;
        button1.Enabled = true;
    }
```

Infine clicchiamo due volte sul bottone "estrai" e gestiamo la procedura di estrazione

```
private void button1_Click_1(
                          object sender, EventArgs e)
{
  //controlliamo l'esistenza del file... meglio
  //se si abilita la condizione anche sul
  //pulsante
  if (file == null || file == string.Empty) return;
  if (!File.Exists(file)) return;

     DialogResult result =
                folderBrowserDialog1.ShowDialog();
  if (result != DialogResult.OK) return;
      string PathTempWorkingFolder =
           folderBrowserDialog1.SelectedPath+"\\";

     System.IO.FileStream streamWriter;
     //otteniamo lo stream di input dal file
     ZipInputStream s = new ZipInputStream(
                     System.IO.File.OpenRead(file));
     ZipEntry theEntry;

  if (!System.IO.Directory.Exists(
                         PathTempWorkingFolder))
                  //pathWorkingFolder+"\\temp"))
     System.IO.Directory.CreateDirectory(
                          PathTempWorkingFolder);
     int num = 0;
  while ((theEntry = s.GetNextEntry()) != null)
  {
      if (isSelected(theEntry.Name))
      {
          if (File.Exists(PathTempWorkingFolder
              + theEntry.Name) && MessageBox.Show(
            "Il file " + PathTempWorkingFolder + "\\" +
            theEntry.Name + " già esiste nella cartella,
                       sovrascriverlo?", "Info",
                      MessageBoxButtons.YesNo,
                      MessageBoxIcon.Question)
                           == DialogResult.No)
            continue;
            num++;
            streamWriter = System.IO.File.Create(
                          PathTempWorkingFolder
                               + theEntry.Name);
            int size = 2048;
            byte[] data = new byte[2048];
          while (true)
```

```
        {
            size = s.Read(data, 0, data.Length);
            if (size > 0)
            {
                streamWriter.Write(data, 0, size);
            }
            else
             {
                  break;
             }
        }

        streamWriter.Close(); streamWriter = null;
      }

    }
    s.Close(); s = null; theEntry = null;
    MessageBox.Show("Estratti " + num + " file",
                    "Info", MessageBoxButtons.OK,
                     MessageBoxIcon.Information);
    GC.Collect();
}
```

Scriviamo il metodo *isSelected* che controlla se una entry del file deve essere realmente estratta

```
private bool isSelected(string entryName)
{
    foreach (string stringa in listBox2.Items)
    {
        if (String.Compare(stringa, entryName, true)
                                   ==0) return true;
    }
    return false;
}
```

COME FUNZIONA

La pressione sul menu open apre una dialog box e riempie il contenuto della variabile privata "file" con il nome del file compresso su cui vogliamo agire, a questo punto passa il controllo al metodo zipinfo(). Questo metodo effettua un ciclo di while sulle entry che compongono il file in questione e riempie la listbox1. La pressione sul secondo pulsante sposta i nomi dei file contenuti in listbox1 da questa a listbox2. La pressione del tasto estrai effettua di nuovo un ciclo sul file e se una delle entry viene trovata anche in listbox2 allora la estrae.

FACCIAMOLO IN VISUAL BASIC

I passi da uno a 5 rimangono identici a quelli della procedura C#, la dichiarazione della variabile che conterrà il nome del file da controllare diventerà

```
Private myfile As String
```

clicchiamo sulla voce *"Open"* per creare il codice di gestione dell'evento, che diventerà

```
Private Sub OpenToolStripMenuItem_Click(ByVal
                    sender As System.Object, ByVal e As
                           System.EventArgs) Handles
                          OpenToolStripMenuItem.Click
    Dim result As DialogResult =
                         OpenFileDialog1.ShowDialog()

    If (result = windows.Forms.DialogResult.OK)
                                                 Then
      myfile = OpenFileDialog1.FileName
      zipinfo()
    End If

  End Sub
```

GLI IMPORT DA UTILIZZARE IN VISUAL BASIC

È necessario aggiungere nella classe le seguenti linee

```
Imports ICSharpCode.SharpZipLib
Imports ICSharpCode.SharpZipLib.Zip
Imports ICSharpCode.SharpZipLib
                  .Zip.Compression
Imports ICSharpCode.SharpZipLib
                  .Zip.Compression.Streams
Imports System.IO
```

Scriviamo il metodo *zipinfo()*

```
Private Sub zipinfo()
    ListBox1.Items.Clear()
    ListBox2.Items.Clear()
    Button2.Enabled = False
    Button1.Enabled = False
    Dim s As ZipInputStream = New
          ZipInputStream(File.OpenRead(myfile))
    Dim theEntry As ZipEntry
    Dim pathproject As String = String.Empty
    Dim pathimg As String = String.Empty
    theEntry = s.GetNextEntry
    Do While Not theEntry Is Nothing
      ListBox1.Items.Add(theEntry.Name)
      theEntry = s.GetNextEntry
    Loop
    s.Close()
    s = Nothing
    theEntry = Nothing
  End Sub
```

Gestiamo l'evento *SelectedIndexChanged*

```
button2.Enabled = true;
```

Scriviamo il codice di gestione relativo all'evento

onclick sul pulsante che vogliamo usare per selezionare i file da estrarre

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles Button1.Click
    If ListBox1.SelectedIndex < 0 Then
        Return
    End If

    ListBox2.Items.Add(ListBox1.SelectedItem)
    ListBox1.Items.Remove(ListBox1.SelectedItem)
    Button2.Enabled = False
    Button1.Enabled = True
End Sub
```

GLI IMPORT DA UTILIZZARE IN C#

E' necessario aggiungere nella classe le seguenti linee

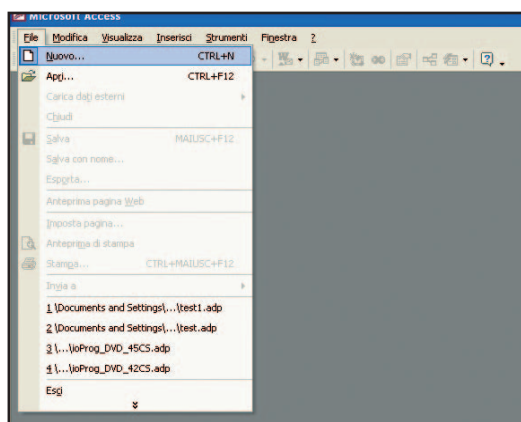
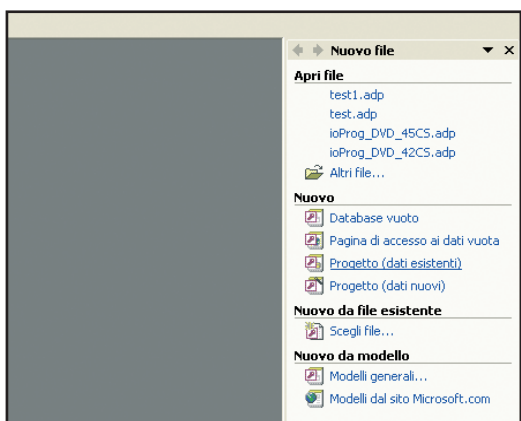
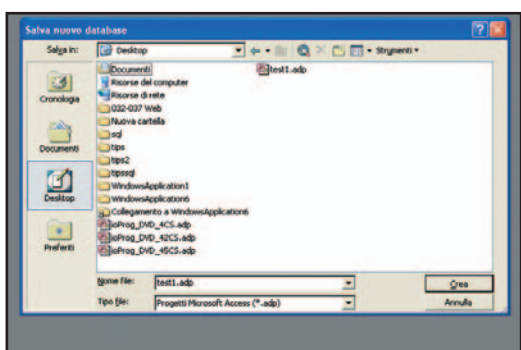
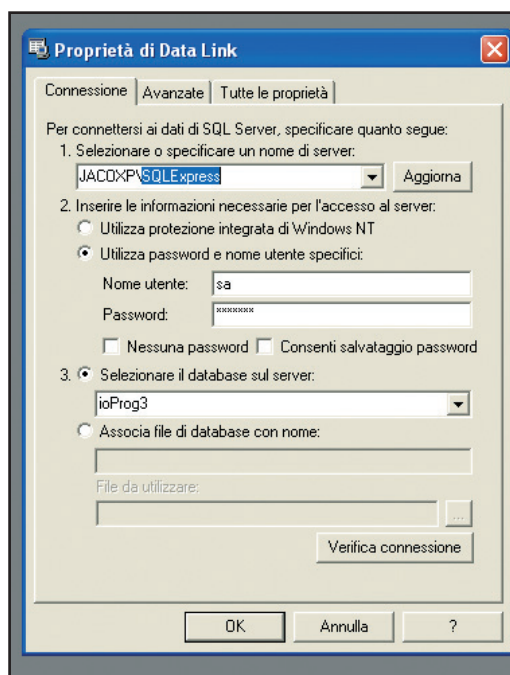
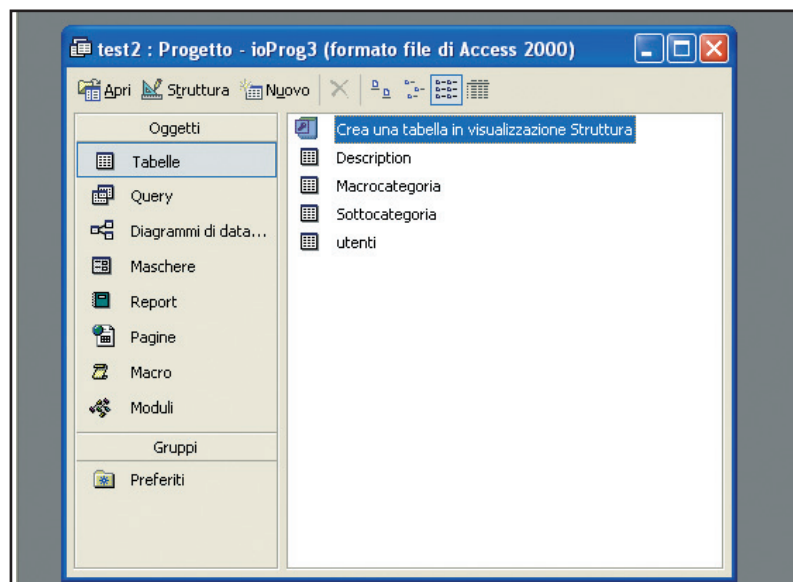
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ICSharpCode.SharpZipLib.Zip;
using ICSharpCode.SharpZipLib.Zip.Compression;
using ICSharpCode.SharpZipLib.Zip.Compression
                                        .Streams;
using System.IO;
```

Scriviamo il codice relativo al bottone estrai

```
Private Sub Button2_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles Button2.Click
    'controlliamo l'esistenza del file... meglio se si
            abilita la condizione anche sul pulsante
    If (myfile Is Nothing Or myfile Is
                            String.Empty) Then
        Return
    End If
    If Not (File.Exists(myfile)) Then
        Return
    End If
    Dim result As DialogResult =
                FolderBrowserDialog1.ShowDialog()
    If Not (result =
            Windows.Forms.DialogResult.OK) Then
        Return
    End If

    Dim PathTempWorkingFolder As String =
            FolderBrowserDialog1.SelectedPath + "\"
    Dim streamWriter As System.IO.FileStream
    'otteniamo lo stream di input dal file
    Dim s As ZipInputStream = New
        ZipInputStream(System.IO.File.OpenRead(
                                        myfile))
    Dim theEntry As ZipEntry
    If Not (System.IO.Directory.Exists(
                    PathTempWorkingFolder)) Then
        System.IO.Directory.CreateDirectory(
                            PathTempWorkingFolder)

    End If
    Dim num As Integer = 0
    theEntry = s.GetNextEntry()
    Do While Not theEntry Is Nothing
        If (isSelected(theEntry.Name)) Then
            streamWriter = System.IO.File.Create(
            PathTempWorkingFolder + theEntry.Name)
            Dim size As Integer
            Dim mydata As Byte()
            Do While (True)
                size = s.Read(mydata, 0,
                                    mydata.Length)
                If (size > 0) Then
                    streamWriter.Write(mydata, 0, size)
                Else
                    Stop
                End If
                streamWriter.Close()
                streamWriter = Nothing
            Loop

            num = num + 1
        End If
        theEntry = s.GetNextEntry()
    Loop
    s.Close()
    s = Nothing
    theEntry = Nothing
    MessageBox.Show(
                "Estratti " + num + " file",
                "Info", MessageBoxButtons.OK,
                MessageBoxIcon.Information)
    GC.Collect()
End Sub
Private Function isSelected(
        ByVal entryName As String) As Boolean
    Dim stringa As String
    For Each stringa In ListBox2.Items
        If (String.Compare(stringa, entryName,
                                True) = 0) Then
            Return True
        End If
    Next

    Return False

End Function
```

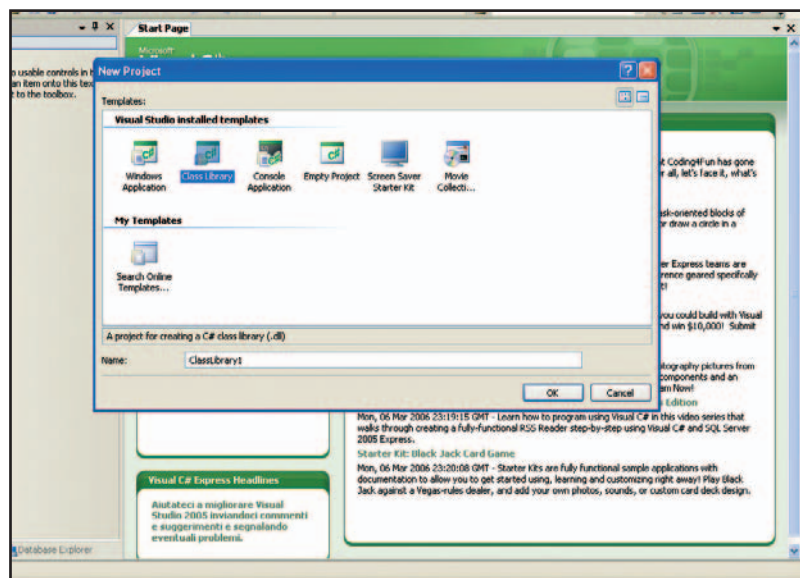
COME POSSO USARE ACCESS COME CLIENT DI SQL SERVER 2005?

PUÒ ESSERE UNA SOLUZIONE MOLTO COMODA PER CREARE RAPIDAMENTE DELLE MASCHERE O GESTIRE UN DATABASE IN MODALITÀ STRUTTURA O INSERIMENTO DATI. LA PROCEDURA È SEMPLICISSIMA

1 Prima di tutto è necessario avviare access e scegliere file/nuovo

2 Dal menu laterale scegliete "Progetto dati Esistente" se volete utilizzare un db già presente in SQL Server, altrimenti "Progetto Dati Nuovi" se volete creare un nuovo db su SQL Server

3 Scegliete la posizione in cui salvare il file di progetto

4 Inserite le impostazioni di connessione, avendo cura di aggiungere il nome dell'istanza di SQLServer a cui connettersi immediatamente dopo il suo nome preceduta dal simbolo "\" senza le virgolette

5 A questo punto potete lavorare sulle tabelle in modo consueto come siete abituati a fare in un progetto Access

COME POSSO SCRIVERE UNA STORED PROCEDURE IN MANAGED CODE?

A PARTIRE DA VISUAL STUDIO 2005 E SQL SERVER 2005 È POSSIBILE SCRIVERE UNA SP DIRETTAMENTE DAL CLR, QUESTO CONSENTE DI POTER UTILIZZARE UN UNICO AMBIENTE PER TUTTE LE OPERAZIONI

FACCIAMOLO IN C#

1 Creiamo un nuovo progetto e scegliamo “Class Library” e diamo al progetto un nome significativo, ad esempio: SQLServerHostTest

2 sostituiamo il nome della classe base con uno di nostro gradimento

```
namespace SQLServerHostTest
{
   public class HostFunctions
   {
   }
}
```

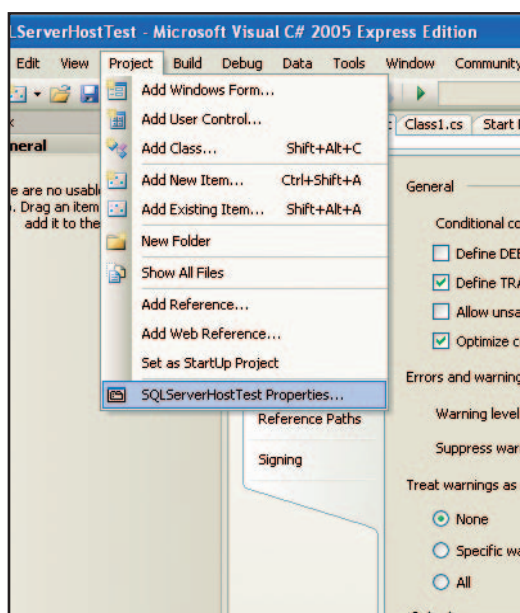
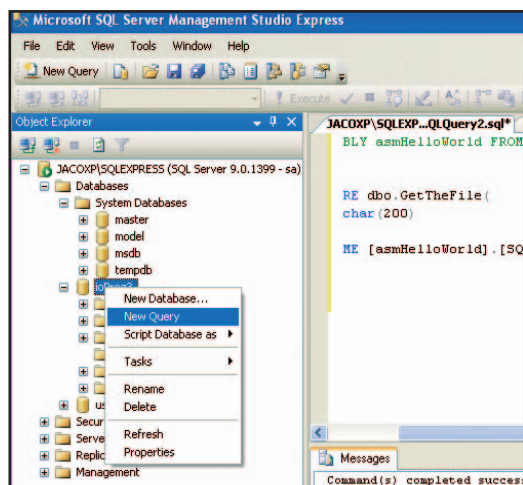
GLI IMPORT DA USARE IN C#

Nella prima parte della classe è necessario aggiungere

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using Microsoft.SqlServer;
using Microsoft.SqlServer.Server;
```

3 Aggiungiamo la funzione che sarà “Hostata” in SQL Server

```
namespace SQLServerHostTest
{
   public class HostFunctions
   {
[Microsoft.SqlServer.Server.SqlProcedure()]
public static void GetDescription(string nome)
{
    using (SqlConnection connection = new
         SqlConnection("context connection=true"))
    {
      connection.Open();
      SqlCommand command = new SqlCommand(
               "SELECT * FROM Description " +
           "WHERE NomeFile like '%"+ @nome
                        +"%'", connection);
      command.Parameters.AddWithValue(
                          "@nome", nome);
      SqlContext.Pipe.ExecuteAndSend(
                               command);
      }
    }
  }
}
```

4 Clicchiamo sul menu “Project SqlServerHost-Test Properties” e settiamo il path dove vogliamo che il compilatore salvi la nostra nuova dll. Quando siamo certi che tutto sia idoneo ai nostri scopi compiliamo il progetto tramite il tasto F6

5 Avviamo SQL Server Management Studio e clicchiamo con il tasto destro sul database che vogliamo usare come test, selezionando “New Query”

6 Nella finestra della query digitiamo

```
CREATE ASSEMBLY asmHelloWorld FROM
                        'c:\SqlServerHostTest.dll'
```

7 Eseguiamo la query tramite il bottone execute, cancelliamola e scriviamone un'altra

```
CREATE PROCEDURE dbo.GetTheFile(
                        @nome as nvarchar(200) )
AS EXTERNAL NAME [asmHelloWorld].[
            SQLServerHostTest.HostFunctions].[
                                GetDescription];
```

8 Proviamo la nuova stored procedure con

```
USE [ioProg3]
GO
DECLARE@return_value int
EXEC    @return_value = [dbo].[GetTheFile]
                @nome = N'Flash'
SELECT  'Return Value' = @return_value
GO
```

COME FUNZIONA?

Abbiamo creato una DLL al cui interno è "immagazzinata" una query. In SQL Server abbiamo registrato la DLL come assembly esterno. Abbiamo poi creato una procedure il cui codice non è espresso da una funzione programmatica all'interno di SQL ma è prelevato direttamente dall'assembly.

FACCIAMOLO IN VISUAL BASIC

Tutti i passi rimangono identici ai precedenti, cambia esclusivamente il codice da implementare nel progetto, che diventa:

```
amespace SQLServerHostTest1
    Public Class HostFunctions
        <Microsoft.SqlServer.Server.SqlProcedure()> _
        Public Shared Sub GetDescription(ByVal nome
                                        As String)
            Dim command As SqlCommand
            ' Connect through the context connection
            Using connection As New SqlConnection(
                            "context connection=true")
                connection.Open()
                command = New SqlCommand( _
                    "SELECT * from description " & _
                    "WHERE NomeFile like '%" &
                            "@nome" & "%'", connection)
                command.Parameters.AddWithValue(
                                    "@nome", nome)
                ' Execute the command and send the
                            results directly to the client
                SqlContext.Pipe.ExecuteAndSend(
                                        command)
            End Using
        End Sub
    End Class
End Namespace
```

GLI IMPORT DA USARE IN VB.NET

Nella prima parte della classe è necessario aggiungere

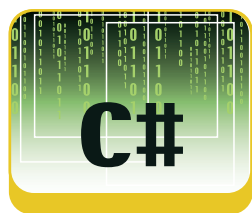
```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using Microsoft.SqlServer;
using Microsoft.SqlServer.Server;
```

LA TABELLA ESISTE?

Sfruttiamo il controllo sugli errori per capire se l'oggetto esiste

```
Private function TabellaPresente
        (TableName as String) As Boolean
        Dim strConn As String
        Dim rs As Recordset
        strConn = "Provider = Microsoft.Jet
OLEDB.4.0; Data Source =
"&_"C:\esempio.mdb"
Set rst=New Recordset
On Error Resume Next
rst.Open tableName, strConn, _
            adOpenForwardOnly,
adLockReadOnly,adCmdTable
TabellaPresente = Not Cbool(Err.Number)
            On Error GoTo 0
End Function
```

INSTANT MESSAGING CON MSN E .NET

DOTMSN È UNA LIBRERIA OPEN SOURCE PER UTILIZZARE LE FUNZIONALITÀ DI MSN MESSENGER IN UNA QUALUNQUE APPLICAZIONE .NET, SIA WINDOWS CHE WEB VEDIAMO COME DOTARE IL NOSTRO SOFTWARE DI UNA CHAT ONE TO ONE

MSN Messenger è un client di instant messaging per la piattaforma Windows, così diffuso che i suoi utenti lo chiamano semplicemente MSN, oppure il Messenger. Esiste anche una versione web dell'applicazione, chiamata MSN Web Messenger. MSN Messenger è basato sul protocollo MSNP, che sta per Mobile Status Notification Protocol, ed è basato su TCP, o eventualmente su http quando è necessario utilizzare un proxy. Il protocollo MSNP si connette al servizio MSN Messenger sulla porta 1863 di messenger.hotmail.com, ed è attualmente giunto alla versione 12 (MSNP12) che corrisponde alla versione 7.5 del client, mentre il successivo, attualmente in beta ad invito, chiamato Windows Live Messenger, utilizzerà la versione 13. Il protocollo MSNP non è di pubblico dominio, a parte le prime versioni pubblicate da Microsoft in un Internet Draft. Dunque molti utenti si sono mes si a sniffare il traffico prodotto durante una sessione MSN e, a colpi di riverse engineering, hanno pubblicato versioni non ufficiali del protocollo. Attualmente la fonte più autorevole, dove trovare informazioni sul formato dei comandi MSNP, è il sito web *http://www.hypothetic.org/docs/msn/index.php*.

DOTMSN

Studiando il protocollo MSNP è possibile creare un'applicazione per autenticarsi al servizio, aprire conversazioni con gli utenti registrati, ed effettuare tutte le operazioni permesse da MSN Messenger. Il compito non sarebbe impossibile, ma non è nemmeno così semplice. Per fortuna esistono librerie che implementano il protocollo, fra le quali dotMSN, che fra l'altro adesso è anche open source e quindi è possibile studiare il codice sorgente. Se invece si vuole solo sfruttare la libreria per integrare in un'applicazione le funzionalità di instant messaging di MSN Messenger, basta utilizzare la libreria dotMSN e magari la completa documentazione fornita a corredo, cosa che faremo in questo articolo. DotMSN non necessita della presenza del client MSN Messenger installato, in quanto si connette direttamente all'host del servizio, e colloquia con il suo protocollo. L'unica pecca, al momento, è il supporto giunto alla versione MSNP9, ma per le funzionalità fondamentali è più che sufficiente.

UN PROGETTO CON DOTMSN

Per scrivere un'applicazione che utilizzi dotMSN, utilizzeremo Visual Studio .NET 2005, ma la versione 2.0 di dotMSN funziona senza nessuna differenza anche con il .NET framework 1.1. È necessario innanzitutto scaricare dotMSN dal sito indicato in bibliografia, ma presente anche sul CD allegato alla rivista, e dopo aver creato un'applicazione Windows, in questo caso C#, aggiungere la libreria fra i riferimenti del progetto stesso. Dunque basta cliccare con il tasto destro su *References* (*Riferimenti* nella versione italiana) e dal menu contestuale scegliere la voce *Add Reference (Aggiungi Riferimento)*. A questo punto basta sfogliare il file system, e ricercare l'assembly *XihSolutions.DotMSN.dll* (**Figura 1**).

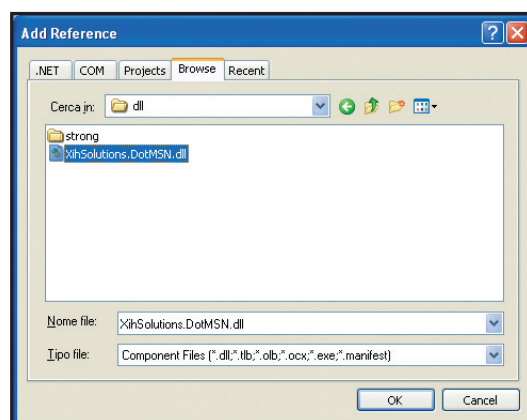



Fig. 1: **Aggiungere il riferimento a DotMSN**

La libreria DotMSN apparirà nel solution explorer di Visual Studio 2005 fra i riferimenti del progetto (**Figura 2**).

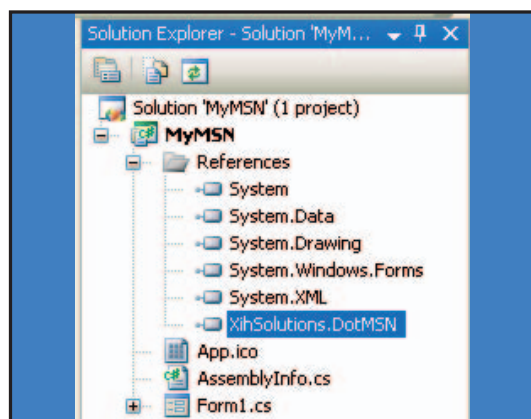



Fig. 2: ***Riferimento all'assembly DotMSN***

LA CLASSE MESSENGER

La classe Messenger di DotMSN è forse la classe principale del pacchetto, in quanto fornisce il punto d'ingresso per programmare un client MSN. La classe Messenger è una classe façade che nasconde tutte le astrazioni di basso livello, ad esempio il processamento dei messaggi in arrivo e in uscita, la gestione del protocollo MSNP, e così via, utilizzando una gestione ad eventi in maniera da informare le classi che sono interessati ad essi. In tal modo i programmatori potranno semplicemente sottoscrivere un evento della classe e gestirlo come meglio credono. Inoltre tramite la lettura di semplici proprietà si potrà verificare lo stato del client, ad esempio la proprietà *Connected* restituisce true se c'è una connessione al server MSN, mentre *Owner,* che deriva dalla classe *Contact,* rappresenta l'utente connesso, di cui si può ricavare il nome visualizzato, leggendo la proprietà *Name,* o tutte le altre informazioni sull'utente:

```
if(messenger.Connected)
{ //connesso al servizio
   string name=messenger.Owner.Name;
   MessageBox.Show("Accesso avvenuto come "+name);
}
```

Nella nostra applicazione di esempio partiremo dunque proprio dall'aggiungere un campo di classe *Messenger* alla form di avvio dell'applicazione.

CONNESSIONE AL SERVIZIO

La prima funzionalità necessaria al funzionamento dell'applicazione è naturalmente quella che effettua l'accesso al servizio. Dunque implementiamo una Form che apparirà subito sopra la barra delle applicazione alla pressione della voce *Accedi* del menu, come in **Figura 3**.

La gestione dei valori inseriti nelle TextBox viene fatta alla chiusura della form stessa, leggendoli dalla classe principale *LaunchForm,* se si è premuto il tasto *Accedi.* Ciò è possibile impostando la proprietà *DialogResult* del pulsante al valore *DilogResult.OK,* mentre al contrario il valore della proprietà per il pulsante *Annulla* sarà impostato a *DialogResult .Cancel.* In tal modo per gestire l'accesso potremo scrivere un metodo come il seguente:

```
ConnectForm connect = new ConnectForm(
messenger.Credentials.Account,
                          messenger.Credentials.Password);
connect.Parent = null;
if (connect.ShowDialog() == DialogResult.OK)
{ if (messenger.Connected)
  {   SetStatus("Disconnessione in corso...");
      messenger.Disconnect(); }
  messenger.Credentials.Account = connect.Username;
  messenger.Credentials.Password = connect.Password;
  SetStatus("Connessione in corso...");
  messenger.Connect();
}
```

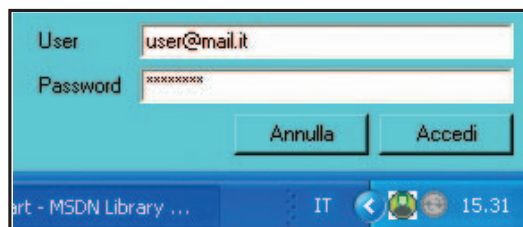
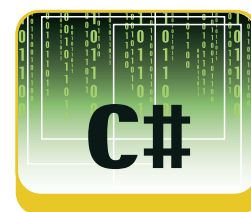



Fig. 3: ***La form di accesso al servizio***

Come potrete notare, la prima condizione da controllare è se si è già connessi al servizio, tramite la proprietà *Connected,* ed in caso affermativo disconnettersi. Per la connessione invece si impostano i valori di *Account* e *Password* leggendoli dalla form di connessione e subito dopo invocando il metodo *Connect.* Ma cosa avviene se la password è errata? Oppure se non è possibile comunque connettersi, o ancora se viceversa la connessione avviene positivamente? È necessario gestire gli eventi della classe *Messenger* e delle sue proprietà.

GLI EVENTI DI MESSENGER

Facciamo un passo indietro e vediamo la sottoscrizione degli eventi della classe *Messenger* che ci serviranno a capire cosa sta succedendo.

```
messenger = new Messenger();
messenger.Credentials.ClientID = "mymsn@mymsn";
messenger.Credentials.ClientCode = "MyMSN-0.1";
messenger.Nameserver.SignedIn += new
                     EventHandler(Nameserver_SignedIn);
```

CLIENTID E CLIENTCODE

Il client identifier ed il client code, permettono di dare una firma identificativa al client connesso alla rete MSN. In questo modo le software house possono avere un'identificazione del proprio prodotto, chiedendo ufficialmente a Microsoft una licenza contenente i due valori. Per emulare il client *Microsoft MSN Messenger* è possibile usare come *ClientID* msmsgs@msnmsgr.com e come *ClientCode* *Q1P7W2E4J9R8U3S5.*

E SE C'È IL PROXY?

È possibile impostare tutti i parametri relativi al proxy da voi utilizzato, grazie alle proprietà *Messenger.Connectivity* che restituisce un oggetto *ConnectivitySettings.* Ad esempio le proprietà per impostare nome del proy, porta, username e password sono rispettivamente *ProxyHost, ProxyPort, ProxyUsername* e *ProxyPassword.*

```
messenger.Nameserver.SignedOff += new
        SignedOffEventHandler(Nameserver_SignedOff);
messenger.Nameserver.AuthenticationError +=
        new.HandlerExceptionEventHandler(
        Nameserver_AuthenticationError);
messenger.NameserverProcessor.ConnectionEstablish
        ed += new EventHandler(
        NameserverProcessor_ConnectionEstablished);
messenger.NameserverProcessor.ConnectionClosed
        += new EventHandler(
        NameserverProcessor_ConnectionClosed);
messenger.NameserverProcessor.ConnectingException
        += new ProcessorExceptionEventHandler(
        NameserverProcessor_ConnectingException);
messenger.NameserverProcessor.ConnectionException
        += new ProcessorExceptionEventHandler(
        NameserverProcessor_ConnectionException);
messenger.ConversationCreated += new
        ConversationCreatedEventHandler(
        messenger_ConversationCreated);
```

Tramite le proprietà *ClientID* e *ClientCode* si può dare una firma al nostro client, per ulteriori informazioni vedi il box relativo. Successivamente sottoscriviamo alcuni eventi necessari a controllare il funzionamento del client ed il suo stato. La proprietà *Nameserver* di Messenger restituisce un *NSMessageHandler,* cioè un oggetto che si occupa di gestire il protocollo MSNP. L'evento *SignedIn* si verifica quando è terminata la fase di autenticazione ed il server ha dato autorizzazione all'accesso, quindi da questo momento si è connessi alla rete MSN. Il metodo che gestisce l'evento è il seguente:

```
void Nameserver_SignedIn(object sender, EventArgs e)
{ SetStatus("Online: " + messenger.Owner.Name);
  messenger.Owner.Status = PresenceStatus.Online;
  notifyIcon.ShowBalloonTip(4000, "Accesso
        avvenuto", "Effettuato l'accesso come
        "+messenger.Owner.Name, ToolTipIcon.Info);
  EnableMenuItem(false, accediToolStripMenuItem);
  EnableMenuItem(true, contattiToolStripMenuItem);
  Invoke(new UpdateContactListDelegate(
        UpdateContactlist));
}
```

Oltre ad operazioni puramente grafiche, come impostare la label che indica lo stato di connessione, mostrare un *BalloonTip*, ed abilitare i menu che è possibile utilizzare da questo momento, il metodo imposta la proprietà *Owner.Status* al valore *PresenceStatus.Online,* in questo modi nostri contatti potranno accorgersi di noi. L'elenco di contatti viene ottenuto a questo punto tramite il metodo *UpdateContactList,* chiamto tramite Invoke ed un delegate perché è necessario eseguirlo nel thread della nostra applicazione e non in quello della classe che genera l'evento *SignedIn.* Al contrario *SignedOff* è un evento che si verifica alla disconnessione, ed il metodo *Nameserver_SignedOff* non fa altro che mostrarci un messaggio per avvisarci di ciò, e della ragione della disconnessione.
Infatti il metodo ha la firma

```
void Nameserver_SignedOff(
        object sender, SignedOffEventArgs e)
```

L'oggetto *SignedOffEventArgs* possiede una proprietà *SignedOffReason*, che può assumere uno dei valori dell'omonima enumerazione, vale a dire None, che avviene generalmente se siamo noi a chiedere la disconnessione, o se ad esempio cade la connessione alla rete, *OtherClient* se la connessione al servizio con lo stesso utente è avvenuta da un altro client, *ServerDown* se è il server è andato giù. Se la coppia *Username-Password* non è valida, si ha naturalmente un errore di autenticazione, gestibile dall'evento *AuthenticationError.* La proprietà *NameserverProcessor* di *Messenger,* restituisce invece un oggetto *NSMessageProcessor,* classe che si occupa di gestire l'IO con il *Notification Server,* cioè il server che gestisce i messaggi ed il loro contenuto. La classe *NSMessageProcessor* deriva dalla *SocketMessageProcessor,* della quale andremo a gestire gli eventi di connessione e disconnessione. Infine gestiremo l'evento *ConversationCreated* della classe *Messenger,* evento che si verifica all'avvio di una conversazione con un contatto, sia iniziata localmente, che per un invito remoto, torneremo sull'argomento nel paragrafo relativo alle conversazioni.

LA LISTA DEI CONTATTI

Una volta connessi posiamo ottenere dal server del servizio MSN la lista dei nostri contatti, aggiungendoli come voci di sottomenu del menu principale *Contatti.*

```
private void UpdateContactlist()
{  if (messenger.Connected == false)
   return;
   contattiToolStripMenuItem.DropDownItems.Clear();
   ToolStripMenuItem item = new ToolStripMenuItem();
   foreach (Contact contact in messenger.ContactList.All)
   {  item = new ToolStripMenuItem();
      item.Text = contact.Name;
      item.Tag = contact;
      contact.ContactOnline += new
            Contact.ContactChangedEventHandler(
            contact_ContactOnline);
      item.Click += new EventHandler(item_Click);
      contattiToolStripMenuItem.DropDownItems.Add(
            item); }
}
```

DOTMSN

DotMSN è la libreria per sfruttare le funzioni di _MSN Messenger_ nelle applicazioni .NET, è open source ed è gratuita, ed inoltre è liberamente utilizzabile sia in applicazioni commerciali che non commerciali. L'indirizzo web da cui è possibile effettuare il download è http://www.xihsolutions.net/dotmsn/download.html.

IL PROGETTO

Sul CD o sul sito web di IoProgrammo trovate il codice completo dell'applicazione. Esso è costituito da una soluzione per Visual Studio 2005, ma naturalmente è possibile compilare da riga di comando per mezzo del compilatore csc fornito con il framework .NET 2.0

per ottenere la lista dei nostri contatti basta utilizzare la proprietà *ContactList.All* che restituisce un enumeratore, utilizzabile in un ciclo *foreach*. Per ogni oggetto *Contact* della lista andremo a gestire l'evento *ContactOnline*, per sapere se e quando il contatto ha effettuato l'accesso. Il metodo che gestisce l'evento è il seguente:

```
void contact_ContactOnline(object sender, EventArgs e)
{   Contact contact = sender as Contact;
    if (contact != null)
    {   notifyIcon.ShowBalloonTip(3000, "Contatto
            online", contact.Name+" ha effettuato
                l'accesso", ToolTipIcon.Info); }
}
```

L'oggetto *Contact* è ottenuto dall'oggetto *sender*, di cui possiamo ottenere informazioni come il nome visualizzato, la mail dell'account, lo stato, e così via. In questo caso mostriamo un messaggio con lo *Screen Name* del nostro contatto.

AVVIARE UNA CONVERSAZIONE

Una conversazione può essere iniziata localmente, invitando un contatto, oppure a partire da un invito remoto di un nostro contatto online. In entrambi i casi si verifica un evento *ConversationCreated*, il cui gestore riceverà come parametro un oggetto *ConversationCreatedEventArgs*. La proprietà *Initiator* di quest'ultimo darà informazioni su chi ha effettuato la richiesta di iniziare una conversazione, se in particolare esso è null, significa che la richiesta proviene da un client remoto.

```
void messenger_ConversationCreated(
        object sender, ConversationCreatedEventArgs e)
{   if (e.Initiator == null)
    { this.Invoke(new CreateConversationDelegate(
            CreateConversationForm), new object[]
                { e.Conversation }); }
}
private delegate ConversationForm
 CreateConversationDelegate(Conversation conversation);
private ConversationForm CreateConversationForm(
                Conversation conversation)
{  ConversationForm conversationForm = new
            ConversationForm(conversation);
   conversationForm.Handle.ToInt32();
   return conversationForm;
}
```

Notate che nel caso di una richiesta remota, la Form di conversazione viene creata, ma non mostrata, fino all'arrivo di un messaggio, ed allora facciamo solo in modo di creare l'handle della finestra da mostrare successivamente. Viceversa, se siamo noi a voler contattare un nostro contatto, cliccando sul relativo item di menu, il metodo invocato sarà il seguente, che crea e mostra la Form.

```
void item_Click(object sender, EventArgs e)
{  ToolStripMenuItem item = sender as
                ToolStripMenuItem;
   if (item != null && item.Tag is Contact)
   {  Contact selectedContact = item.Tag as Contact;
      if (messenger.Connected && selectedContact !=
            null && selectedContact.Online == true)
      {   Conversation conversation =
                messenger.CreateConversation();
          conversation.Invite(selectedContact);
          ConversationForm form = new
                ConversationForm(conversation);
          form.Show(); }
   }
}
```

Dopo aver ricavato il contatto selezionato, viene creata una conversazione, ed inviato il relativo invito al contatto remoto. L'oggetto *Conversation* viene utilizzato per creare la form di conversazione. Il codice completo della Form di conversazione potete osservarlo prendendolo dal CD allegato, mostriamo però come inviare un messaggio al contatto:

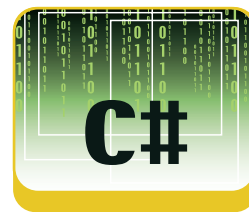
```
TextMessage message = new TextMessage(
                txtInput.Text);
Conversation.Switchboard.SendTextMessage(message);
```

mentre alla ricezione di un messaggio si verificherà l'evento *TextMessageReceived* dal quale si può ricavare il nome del contatto ed il testo del messaggio:

```
private void Switchboard_TextMessageReceived(
        object sender, TextMessageEventArgs e)
{ //...
  string contatto= e.Sender.Name;
  string messaggio= e.Message.Text;
  //mostra messaggio
}
```

CONCLUSIONI

Grazie alla libreria *dotMSN* abbiamo integrato in una applicazione .NET le funzionalità del servizio *MSN Messenger*. *DotMSN* fornisce in maniera semplice la possibilità di accedere al servizio MSN, di utilizzare ogni sua funzionalità come ricavare l'elenco dei contatti, spedire e ricevere messaggi, e così via, ed in più è una libreria free.

Antonio Pelleriti

CONTROL.INVOKE

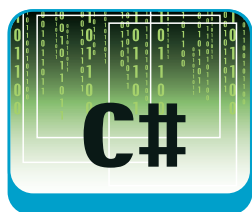
Per eseguire un'operazione sull'interfaccia grafica, ad esempio impostare il tsto di una Label, in un'applicazione multithread, in alcuni casi è necessario utilizzare il metodo *Control.Invoke*. Il caso di cui parliamo è quando l'operazione è effettuata in un thread diverso ad esempio da quello in cui esegue la form che contiene il controllo. È necessario dunque definire un delegate, quindi il metodo che imposta il testo, ed infine eseguire il metodo *Invoke*, ad esempio così:

```
MiaForm.Invoke(new
        MioDelegate(
    EnableMenuItemSafe),
          new object[]
          {arg1,arg2});
```

L'AUTORE

Antonio Pelleriti è ingegnere informatico, e nel tempo libero divulga il verbo dello sviluppo software, in particolare su piattaforma .NET. Potete contattare l'autore per suggerimenti, critiche o chiarimenti all'indirizzo e-mail antonio.pelleriti@ioprogrammo.it, sul forum di IoProgrammo o sul sito web www.dotnetarchitects.it

ALLA SCOPERTA DEL DATAGRIDVIEW

IL .NET FRAMEWORK 2.0 HA INTRODOTTO IL CONTROLLO DATAGRIDVIEW PER LA PRESENTAZIONE DI DATI IN FORMA TABELLARE, UN NOTEVOLE PASSO AVANTI RISPETTO AL VECCHIO DATAGRID

Uno dei modi più comuni di presentare i dati in un'applicazione windows è quello tabellare. Per intenderci il classico modo utilizzato da Excel, dove gli elementi sono rappresentati in una tabella composta da righe e colonne. Questa modalità di visualizzazione è molto usata all'interno delle applicazioni, e anche in quelle sviluppate da noi non mancherà certo la necessità di dover rappresentare una qualche informazione visualizzandola sotto forma di tabella. Per agevolare il compito del programmatore nella presentazione di dati sotto questa forma. Il .NET Framework 1.0 e di seguito l'1.1 mettevano a disposizione il controllo *DataGrid*, una classica griglia composta da righe e colonne. La *DataGrid* consentiva di selezionare, allargare e restringere righe e colonne, ma diventava complicata da gestire quando era necessario personalizzarla per i propri scopi, soprattutto in maniera programmatica. Questo è lo scenario che ha portato alla nascita di un controllo totalmente nuovo, chiamato *DataGridView*.

IL CONTROLLO DATAGRIDVIEW

Il controllo *DataGridView* fornisce numerosi metodi, proprietà, ed eventi per personalizzare il suo aspetto ed il suo comportamento, sia a design time, in modo visuale, da un IDE come Visual Studio, sia a runtime, in maniera programmatica. Spulciando nella documentazione si nota subito come siano state introdotte molte nuove caratteristiche rispetto al tradizionale *DataGrid*. La classe *DataGridView* permette di definire diverse tipologie di colonne, per diversi tipi di dati da trattare. Inoltre esse sono anche estensibili e personalizzabili in maniera più flessibile ed allo stesso tempo semplice di quanto fosse possibile con *DataGrid*. Mentre *DataGrid* era un controllo dedicato alla visualizzazione di elementi provenienti da una sorgente dati, ad esempio una tabella di un database, il controllo *DataGridView* permette di lavorare in maniera trasparente con oggetti eterogenei come ad esempio dati locali o collezioni di *business object* o entrambi in contemporanea. Il controllo *DataGridView* permette molte operazioni per la configurazione dei singoli componenti della griglia. Ad esempio è possibile nascondere righe, colonne, intestazioni, permette di bloccarle per evitarne lo scrolling, consente la visualizzazione di *ToolTip* e menu contestuali per ogni singola cella, riga o colonna, o ancora configurare i bordi, le dimensioni i colori. Una caratteristica presente invece nel vecchio controllo *DataGrid* e non più presente in *DataGridView* è la possibilità di visualizzare dati di due tabelle collegate fra loro, ad esempio immaginare una relazione fra clienti e ordini. Con le *DataGridView* è necessario usare due controlli differenti per implementare una vista *Master/Details*.

IL DESIGNER DI VISUAL STUDIO

Una *DataGridView* è totalmente configurabile utilizzando il designer di Visual Studio 2005. È sufficiente creare una nuova Form e trascinare dalla Toolbox il controllo *DataGridView* sulla sua superficie. Se l'applicazione contiene già una DataSource, essa può essere assegnata alla griglia per mezzo del suo *smart tag* come visualizzato in **Figura 1**.

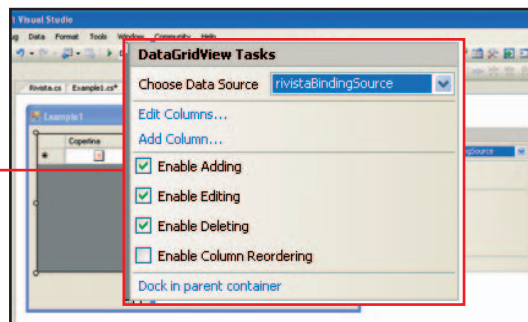



Fig. 1: Lo smart tag della DataGridView

Un'altra possibilità, ancora più immediata, è quella di trascinare la DataSource, ad esempio una *DataTable*, direttamente sulla form. Sempre per mezzo dello *smart tag*, è possibile accedere a diverse opzio-

ni per personalizzare l'aspetto della *DataGridView.* Ad esempio cliccando su *Edit Columns* o *Add Column,* è possibile aggiungere e personalizzare le colonne (**Figura 2**).

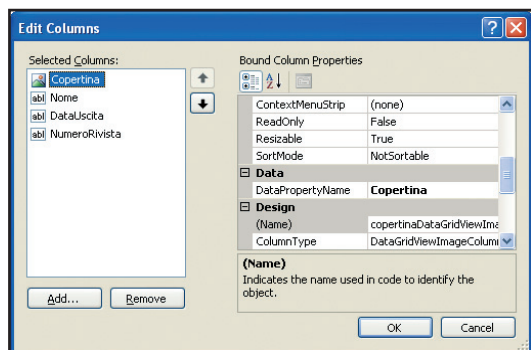



Fig. 2: ***Creazione delle colonne della DataGridView***

Naturalmente se il controllo viene associato ad una sorgente dati, le colonne saranno generate automaticamente, altrimenti è possibile aggiungerne a piacimento cliccando sul pulsante *Add.*

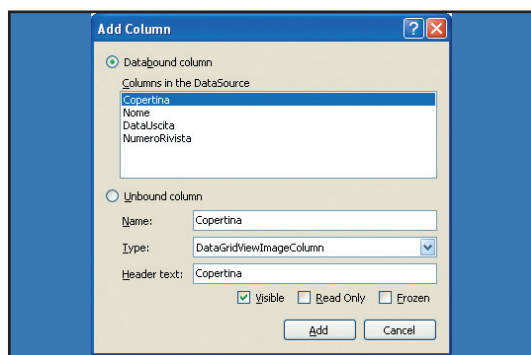



Fig. 3: ***Aggiungere nuove colonne***

Per visualizzare i dati sarà sufficiente impostare la proprietà *DataSource* della classe *DataGridView.*
La classe *DataGridView* supporta il classico modello del data-binding per le *Windows Forms.* E cioè la sorgente dati può essere di un qualunque tipo che implementi una delle interfacce seguenti:

- **List**, ad esempio le liste generiche *List<T>*, e inclusi gli array monodimensionali.
- **ListSource** ad esempio oggetti *DataTable* e *DataSet.*
- **BindingList**, per esempio la classe BindingList.
- **BindingListView** ad esempio la classe *BindingSource.*

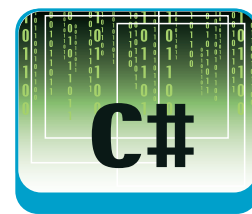
In genere si associa alla proprietà *DataSource* un componente *BindingSource,* il quale a sua volta viene associato alla vera sorgente dati, oppure popolata con collezioni di *business objects.* Il componente *BindingSource* effettua automaticamente il binding ad una grande varietà di sorgenti e risove automaticamente diverse questioni relative al data binding.

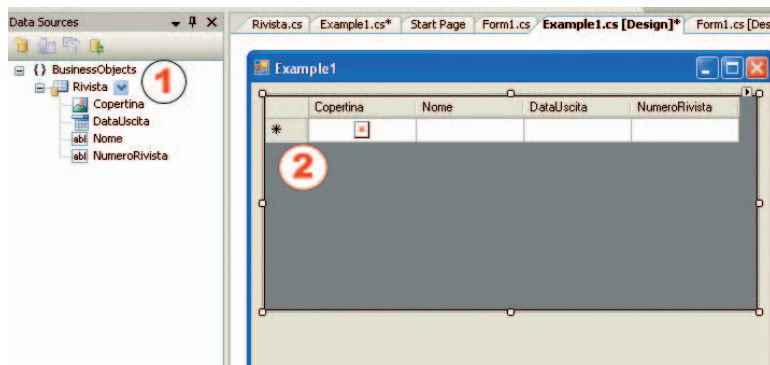
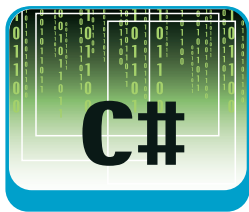
PRIMI PASSI

Partiremo con un esempio semplice, dove utilizzeremo una *List<Rivista>* contenente i dati relativi alle uscite di ioProgrammo, comprensivi dell'immagine della copertina. Questi dati dovranno essere visualizzati in una *DataGridView.* Notate che faremo uso dei *generics*, anche essi introdotti nel framework 2.0. L'esempio è il seguente:

```
List<Rivista> riviste = new List<Rivista>();
Rivista r1 = new Rivista();
r1.Nome = "IoProgrammo";
r1.NumeroRivista = 92;
r1.DataUscita = new DateTime(2005, 6, 1);
r1.Copertina = img92;
Rivista r2 = new Rivista();
r2.Nome = "IoProgrammo";
r2.NumeroRivista = 93;
r2.DataUscita = new DateTime(2005, 7, 1);
r2.Copertina = img93;
riviste.Add(r1);
riviste.Add(r2);
this.dataGridView1.DataSource = riviste;
```

Le righe della *DataGridView* apparirebbero troppo strette per contenere le immagini delle copertine intere, come da nostra intenzione. Utilizzeremo quindi la proprietà *AutoSizeRowsMode* impostandola al valore *DataGridViewAutoSizeRowsMode.AllCells* per autodimensionare sia le celle di intestazione sia quelle contenenti i dati:

```
dataGridView1.AutoSizeRowsMode =
        DataGridViewAutoSizeRowsMode.AllCells;
```

DATASOURCE E DATAGRIDVIEW

È possibile aggiungere una DataGridView automatizzando la creazione di colonne e dei relativi tipi se si ha a disposizione una sorgente dati. Ad esempio, implementando una classe *Rivista*, possiamo cliccare sul menu *Data-> Add New Data Source*, selezionare quindi la classe *Rivista*, e vedremo apparire una nuova sorgente dati fra quelle disponibili. A questo punto basta trascinare la *Data Source* su una *Form* per creare la *DataGridView* relativa.

La nostra *DataGridView*, popolata di riviste con immagini di copertina e date di uscita apparirà come in **Figura 4**:

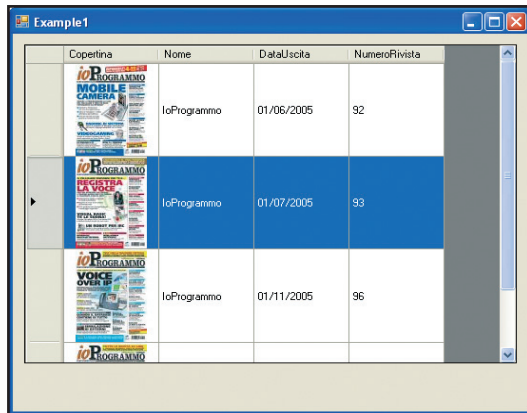



Fig. 4: ***Immagini in una DataGridView***

AGGIUNGERE RIGHE E COLONNE

Il modo più comune di visualizzare i dati nella *DataGridView*, passa dunque per l'uso del *DataBinding*, ciò avviene in particolare se la proprietà *AutoGenerateColumns* è impostata a true, come lo è per default, e congiuntamente le proprietà *DataSource* o *DataMember* vengono impostate o modificate.

È però possibile adottare un approccio programmatico per aggiungere, modificare e rimuovere righe e colonne. Le proprietà *Rows* e *Columns* rappresentano infatti le collezioni di *DataGridViewRow* e *DataGridViewColumn*. In particolare la prima contiene una collection di *DataGridViewCell*, ricavabile dalla proprietà *Cells*. Il seguente esempio aggiunge due colonne, la prima di tipo *DataGridViewTextBoxColumn*, con l'intestazione di colonna impostata mediante la proprietà *HeaderText*.

```
dataGridView1.Columns.Clear();
DataGridViewTextBoxColumn col1 = new
                    DataGridViewTextBoxColumn();
col1.HeaderText = "Text";
```

La seconda colonna viene creata invece a partire da una *DataGridViewLinkCell*, mediante un altro overload del costruttore.

```
DataGridViewLinkCell linkCell = new
                    DataGridViewLinkCell();
DataGridViewColumn col2 = new
                    DataGridViewColumn(linkCell);
col2.HeaderText = "Link";
```

Per aggiungere le colonne alla *DataGridView* è sufficiente utilizzare il metodo *Add* della collection *Columns*:

```
dataGridView1.Columns.Add(col1);
dataGridView1.Columns.Add(col2);
```

Adesso possiamo aggiungere qualche riga, impostando ad esempio il testo contenuto nelle celle di tipo *DataGridViewLinkCell*:

```
for (int i = 0; i < 5; i++)
{
      dataGridView1.Rows.Add();
       dataGridView1.Rows[i].Cells[1].Value =
                         "www.ioprogrammo.it";
}
```

Ogni proprietà, dunque, permette di aggiungere, inserire e rimuovere celle, righe e colonne mediante i classici metodi delle *Collection*, ad esempio *Add*, *Insert*, *Remove*. L'esempio seguente invece crea anche una colonna con una *ComboBox* per scegliere il dato da un elenco discesa. In questo caso la *DataGridViewComboBoxColumn* viene popolata con il nome dei mesi dell'anno, e da questa viene poi creata ed aggiunta una nuova riga alla *DataGridView*.

```
DataGridViewComboBoxColumn comboColumn = new
                    DataGridViewComboBoxColumn();
DataGridViewTextBoxColumn textColumn = new
                    DataGridViewTextBoxColumn();
dataGridView1.Columns.Add(comboColumn);
dataGridView1.Columns.Add(textColumn);
DataGridViewRow row = new DataGridViewRow();
DataGridViewComboBoxCell comboCell = new
                    DataGridViewComboBoxCell();
for (int i = 1; i < 13; i++)
{
comboCell.Items.Add(new DateTime(2000, i, 1
                    ).ToString("MMMM"));
}
comboCell.Value = comboCell.Items[0].ToString();
row.Cells.Add(comboCell);
row.Cells.Add(new DataGridViewTextBoxCell());
dataGridView1.Rows.Add(row);
```

In **Figura 5** viene mostrata una *DataGriView* con la colonna *Combo* appena creata.

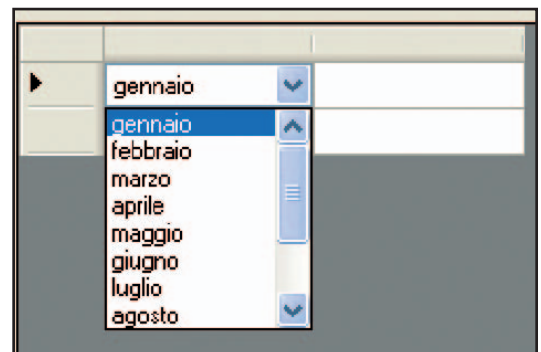



Fig. 5: ***Colonne con ComboBox***

TIPI DI COLONNE

La *DataGridView* utilizza diversi tipi di colonne per mostrare le sue informazioni, e consentirne la modifica.
Come già visto, la proprietà *AutoGenerateColumn* impostata a true consente di creare automaticamente i tipi appropriati ai dati contenuti nella sorgente dati associata.
Qui di seguito riassumiamo i diversi tipi di colonne utilizzabili per i nostri dati, ognuno dei quali è una classe derivata dalla classe *DataGridViewColumn.*

- **IDataGridViewTextBoxColumn**: utilizzata per rappresentare valori visibili come testo, dunque in genere numeri e stringhe.
- **IDataGridViewCheckBoxColumn**: usata con i tipi bool e *CheckState*.
- **IDataGridViewImageColumn**: in genere viene associata ad oggetti *Image* oppure *Icon.*
- **IDataGridViewComboBoxColumn**: quando si vogliono selezionare dei valori da una *ComboBox* per inserirli in una riga. Non viene automaticamente associata alla *DataSource* ma bisogna generarle e popolarle manualmente.
- **IDataGridViewLinkColumn**: permette di mostrare dei link ipertestuali all'interno delle celle, ed anche questa viene associata manualmente ai dati contenuti nella griglia.
- **IDataGridViewButtonColumn**: se si ha bisogno di una colonna con un pulsante mediante il quale eseguire comandi od altre operazioni.

La **Figura 6** mostra un esempio di *DataGridView* con tutti i tipi di colonne appena elencati, creata semplicemente con il designer di Visual Studio 2005 e senza scrivere una sola riga di codice.

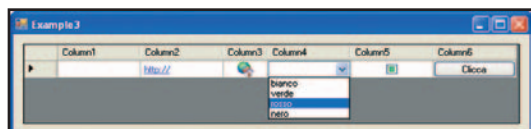



Fig. 6: ***Tutti i tipi di colonne standard***

COLONNE PERSONALIZZATE

Una fra le caratteristiche più apprezzate della nuova DataGridView e quella relativa alla creazione di tipi di colonne personalizzati, se quelli standard non fossero sufficienti, e tutto ciò con la possibilità di ospitare all'interno di una cella un qualsiasi controllo, nuovamente standard o anch'esso personalizzato. Supponiamo di voler permettere all'utente di inserire in una riga della *DataGridView* un nuovo record fra i cui campi è previsto un valore *DateTime.*
Nel primo esempio abbiamo già utilizzato un campo di tipo *DateTime*. Per rappresentare questo tipo di dato le soluzioni possibili sono almeno due. La prima è quella di costringere l'utente ad inserire la data in un certo formato prestabilito, ma pur sempre come string, con tutti i possibili errori di digitazioni volontari o involontari. La seconda soluzione, che utilizzeremo in questo caso, è quella di implementare una colonna che permetta di scegliere una data dal classico controllo *DateTimePicker.*
Partiamo dal mostrare l'architettura che dovrà rispettare la nostra colonna in termini di classi necessarie e delle loro interdipendenze.
Innanzitutto è necessario derivare una nuova classe da *DataGridViewColumn,* che rappresenterà la colonna vera e propria. Una seconda classe dovrà derivare da *DataGridViewCell,* per rappresentare ognuna delle celle visualizzate per ogni riga della colonna. Infine è necessario implementare una classe che derivi da *Control* o dalla classe del controllo che dovrà ospitare, e che implementerà l'interfaccia standard *IDataGridViewEditingControl.*
La **Figura 7** mostra il diagramma delle classi fra le quali potete notare le nostre tre nuove *DateTimePickerColumn*, *DateTimePickerCell* e *DateTimePickerEditingControl*.

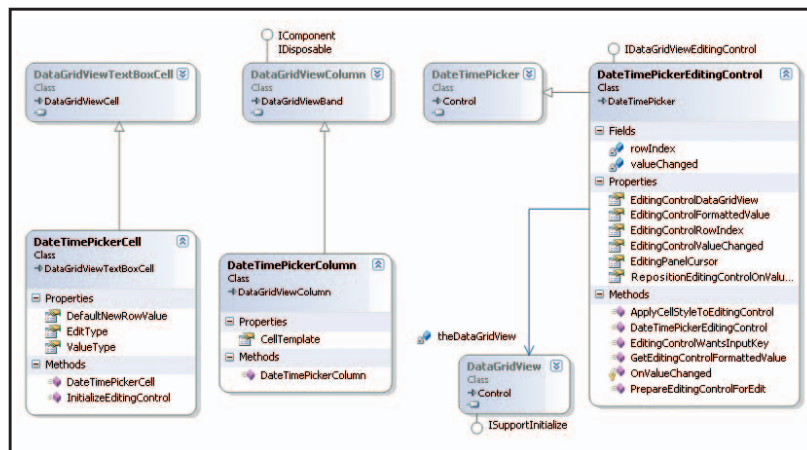
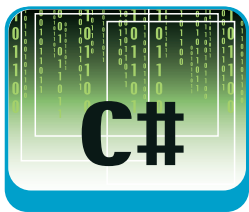



Fig. 7: ***Il diagramma delle classi per la colonna custom***

Chiameremo *DateTimePickerColumn* la classe che estenderà la *DataGridViewColumn.* L'implementazione è la seguente

```
public class DateTimePickerColumn :
                        DataGridViewColumn
{
  public DateTimePickerColumn() : base(new
                        DateTimePickerCell())
  {
  }

  public override DataGridViewCell CellTemplate
  {
    get
    {
```

```
            return base.CellTemplate;
        }
        set
        {
            if (value != null && !value.GetType()
                .IsAssignableFrom(typeof(DateTimePickerCell)))
            {
                throw new InvalidCastException(
                    "La cella deve essere di tipo
                        DateTimePickerCell");
            }
            base.CellTemplate = value;
        }
    }
}
```

La classe che invece dovrà mostrare le celle, deriverà da *DataGridViwTextBoxCell*. Un'implemenrazione che effettua l'override della proprietà *CellTemplate*, in maniera da essere sicuri che esse siano di tipo *DateTimePickerCell*, è la seguente

```
public class DateTimePickerCell :
                                DataGridViewTextBoxCell
{

    public DateTimePickerCell()  : base()
    {
        this.Style.Format = "dd/MMM/yyyy";
    }

    public override void InitializeEditingControl(int
                                          rowIndex, object
    initialFormattedValue, DataGridViewCellStyle
                                    dataGridViewCellStyle)
    {
        base.InitializeEditingControl(rowIndex,
            initialFormattedValue, dataGridViewCellStyle);
        DateTimePickerEditingControl ctl =
                        DataGridView.EditingControl as
                        DateTimePickerEditingControl;
        ctl.Value = (DateTime)this.Value;
    }

    public override Type EditType
    {
        get
        {
            return typeof(DateTimePickerEditingControl);
        }
    }
    public override Type ValueType
    {
        get
        {
            return typeof(DateTime);
        }
    }
```

```
public override object DefaultNewRowValue
    {
        get
        {
            return DateTime.Now;
        }
    }
}
```

Essa definisce il formato da utilizzare per rappresentare la data, il tipo di dati che conterrà la cella, e cioè *DateTime*, il valore di default assegnato alla creazione di una nuova riga. Notate come nel metodo *InitializeEditingControl* e nella proprietà *EditType*, venga definito il tipo del controllo utilizzato come editor dei valori *DateTime* ovvero la classe *DateTimePickerEditingControl*.
Un'implementazione di questa classe è la seguente:

```
class DateTimePickerEditingControl : DateTimePicker,
                            IDataGridViewEditingControl
    {
        private DataGridView theDataGridView;
        private bool valueChanged = false;
        private int rowIndex;

        public DateTimePickerEditingControl()
        {
            this.Format = DateTimePickerFormat.Short;
        }

        public object EditingControlFormattedValue
        {
            get
            {
                return this.Value.ToShortDateString();
            }
            set
            {
                String newValue = value as String;
                if (newValue != null)
                {
                    this.Value = DateTime.Parse(newValue);
                }
            }
        }

        public object GetEditingControlFormattedValue(
            DataGridViewDataErrorContexts context)
        {
            return EditingControlFormattedValue;
        }

        public void ApplyCellStyleToEditingControl(
           DataGridViewCellStyle dataGridViewCellStyle)
        {
            this.Font = dataGridViewCellStyle.Font;
        }
```

BIBLIOTECA

- **CLASSE DATAGRIDVIEW SU MSDN:** http://msdn2.microsoft.com/it-it/library/system.windows.forms.datagridview(VS.80).aspx
- **DATA BINDING WITH WINDOWS FORMS 2.0: PROGRAMMING SMART CLIENT DATA APPLICATIONS WITH .NET** ***Brian Noyes*** **(Addison Wesley Professional)**

```
public int EditingControlRowIndex
{
    get
    {
        return rowIndex;
    }
    set
    {
        rowIndex = value;
    }
}

public bool EditingControlWantsInputKey(
    Keys key, bool dataGridViewWantsInputKey)
{
    switch (key & Keys.KeyCode)
    {
        case Keys.Return:
            return true;
        default:
            return false;
    }
}

public void PrepareEditingControlForEdit(bool
                                    selectAll)
{
}

public bool
        RepositionEditingControlOnValueChange
{
    get
    {
        return false;
    }
}

public DataGridView EditingControlDataGridView
{
    get
    {
        return theDataGridView;
    }
    set
    {
        theDataGridView = value;
    }
}

public bool EditingControlValueChanged
{
    get
    {
        return valueChanged;
    }
    set
    {
            valueChanged = value;
        }
    }

    public Cursor EditingPanelCursor
    {
        get
        {
            return base.Cursor;
        }
    }

    protected override void OnValueChanged(
                                EventArgs eventargs)
    {
        valueChanged = true;
        this.EditingControlDataGridView
                    .NotifyCurrentCellDirty(true);
        base.OnValueChanged(eventargs);
    }
}
```

Nel costruttore viene definito il formato utilizzato dal *DateTimePicker* per le date, mentre la proprietà *EditingControlFormattedValue* permette di ottenere o impostare il valore *DateTime* da visualizzare. Compilato il progetto, provate a creare una nuova *DataGridView*, o ad aggiungere mediante l'editor una nuova colonna ad una esistente. Fra quelle disponibili, ci sarà la colonna appena implementata, e che a runtime vi permetterà di selezionare una data come in **Figura 8**.

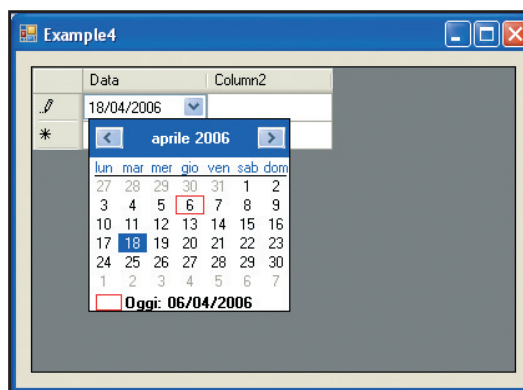



***Fig. 8:* La colonna DateTimePicker**

CONCLUSIONI

Abbiamo visto come utilizzare il nuovo controllo *DataGridView*, visualizzando dati da una sorgente dati, oppure definendo righe, colonne e dati anche programmaticamente. Inoltre la *DataGridView*, ci ha permesso, estendendo classi e implementando interfacce, la realizzazione di un tipo di colonna totalmente personalizzato.

Antonio Pelleriti

CONTATTA L'AUTORE

Potete rivolgere domande e/o chiarimenti o ulteriori richieste all'autore all'indirizzo antonio.pelleriti@ioprogrammo.it, o ancora sul forum di ioProgrammo (http://forum.ioprogrammo.net) o sul sito www.dotnetarchitects.it.

PERSISTENZA IN JAVA2 MICRO EDITION

PREGI E DIFETTI DEL RECORD MANAGEMENT SYSTEM, IL SISTEMA STANDARD PER LA GESTIONE DELLA PERSISTENZA DATI SULLA PIATTAFORMA J2ME. REALIZZIAMO UNA SEMPLICE MIDLET PER LA GESTIONE DI UNA VIDEOTECA PERSONALE

La gestione della persistenza dei dati è senza dubbio un argomento ricorrente durante lo sviluppo di qualsiasi progetto. Le problematiche cambiano in rapporto alla tipologia di applicazione che si sta sviluppando. Un'applicazione per il desktop necessita di attenzioni diverse da una per il Web e, a maggior ragione, un'applicazione per il mondo Mobile è interessata da problemi di persistenza dei dati molto specifici legati all'architettura dei sistemi. Per questa tipologia di applicativi, Java mette a disposizione del programmatore una serie di metodologie la cui scelta dipende dalla tipologia dei dati che si devono trattare e dalle operazioni con le quali questi dati devono essere manipolati. Attraverso la tecnologia J2ME, è possibile leggere dati esterni accedendo allo stream di un file di risorsa presente nel pacchetto jar che contiene l'applicazione stessa. Come si può ben capire, questa tecnica non consente l'inserimento di dati dall'esterno a tempo di esecuzione. Per risolvere questo problema lo strato MIDP (Mobile Information Device Profile) ci viene incontro mettendoci a disposizione il Record Management System (RMS): una semplice base dati record-oriented. L'entità "record" è proprio il punto cruciale di questo sistema. Il tutto è contenuto nel package javax.microedition.rms composto da una serie di interfacce ed eccezioni, su cui ci soffermeremo nel prosieguo dell'articolo, ed una sola classe concreta: RecordStore. Le istanze di tale classe rappresentano degli elementi di persistenza in cui è possibile memorizzare e recuperare i dati.

UN'APPLICAZIONE D'ESEMPIO

In questo articolo realizzeremo una semplice applicazione di esempio che usa il sottosistema RMS per la gestione della nostra videoteca personale.
Tale applicazione, che battezziamo con il nome VideoStore, dovrà mettere a disposizione dell'utente finale le seguenti funzionalità:

1) memorizzazione di un nuovo elemento

Fig.1: *Schermata d'avvio sul Wireless Toolkit Emulator*

2) rimozione di un elemento presente nella videoteca
3) aggiornamento di un elemento precedente memorizzato
4) elenco di tutti gli elementi presenti nella videoteca
5) ricerca di un elemento presente nella videoteca in base ad una data chiave.

In **Figura 2** è illustrato il flusso logico della MIDlet in questione. Questo diagramma, che evidenzia la struttura grafica ed i comportamenti che l'applica-

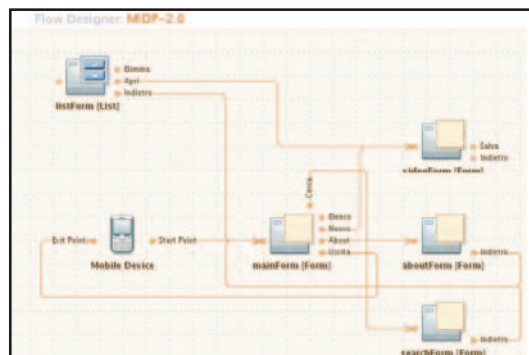



Fig.2: *Struttura della MIDlet per la gestione della videoteca personale*

zione dovrà avere a fronte di determinati comandi, è stato creato con il FlowDesigner presente nel Mobility Pack di Netbeans. Con l'aiuto di quest'ultimo è stata implementata la parte di GUI (Graphical User Interface) dell'esempio, contenuta nella classe it.ioprogrammo.videostore.gui. Non ci soffermeremo oltre su questo aspetto in quanto va al di fuori dello scopo del presente articolo. Come prima cosa si deve definire la classe rappresentante i singoli elementi video, che chiameremo VideoItem.

```
package it.ioprogrammo.videostore.data;
public class VideoItem {
    private int id;
    private String title;
    private String author;
    private int year;
    private String notes;
    public VideoItem() { }
    public int getId() {
        return this.id;}
    public void setId(int id) {
        this.id = id;}
    public String getTitle() {
        return this.title;}
    public void setTitle(String title) {
        this.title = title;}
    public String getAuthor() {
        return author;}
    public void setAuthor(String author) {
        this.author = author;}
    public int getYear() {
        return year;}
    public void setYear(int year) {
        this.year = year;}
    public String getNotes() {
        return notes;}
    public void setNotes(String notes) {
        this.notes = notes;}
}
```

Come possiamo notare, si tratta di una semplice classe conforme alle specifiche javabean che ci permette di gestire, attraverso i metodi setter e getter, le proprietà di un elemento video.

RECORDSTORE: INIZIALIZZAZIONE E APERTURA

Per separare la parte grafica da quella dati e rendere il codice più elegante concentreremo la gestione della persistenza nella classe it.ioprogrammo.videostore.data.DataManager. Per gestire gli elementi di tipo VideoItem useremo un unico oggetto RecordStore. Quest'ultimo viene spesso paragonato, con le dovute limitazioni, ad una tabella di un database. Così come avviene per le tabelle, il nostro contenitore dovrà essere inizializzato prima che sia possibile utilizzarlo. L'invocazione del metodo statico openRecordStore riceve due parametri: il nome del record ed un boolean che, posto a true inizializza il contenitore se necessario.

```
/* Contenitore dati RMS */
private RecordStore videoStore;
/* Costruttore interno per l'implementazione del
                                    pattern singleton */
private DataManager() {
    try {
        // apre il RecordStore se esiste, altrimenti
                                    ne crea uno nuovo
        videoStore = RecordStore.openRecordStore(
                                    "STORE", true);
    } catch (RecordStoreException ex) {
        ex.printStackTrace();}
}
```

È importante evidenziare che, l'invocazione del metodo appena descritto associa il RecordStore alla MIDlet suite che lo crea. Pertanto, due MIDlet suite che inizializzino due RecordStore con lo stesso nome gestiranno due contenitori di persistenza distinti e separati. È inoltre possibile recuperare i nomi di tutti i RecordStore associati ad una MIDlet suite richiamando il metodo statico listRecordStores.

CREATE, READ, UPDATE & DELETE

Adesso ci addentreremo nel cuore della nostra applicazione implementando le funzionalità base, comunemente identificate dall'acronimo CRUD: Create, Remove, Update e Delete di un oggetto di classe VideoItem. Prima di passare al codice è utile capire come i RecordStore gestiscono i dati al loro interno. Come affermato in precedenza questo meccanismo di persistenza è in grado di memorizzare delle entità che chiameremo record. Ogni record è

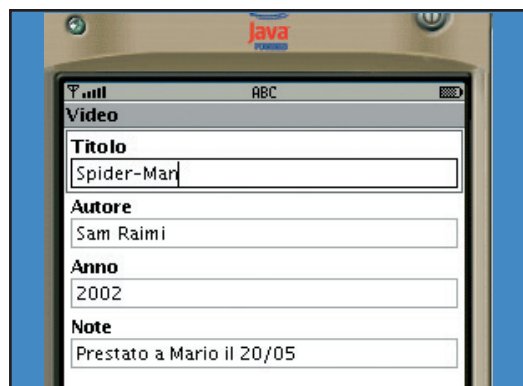
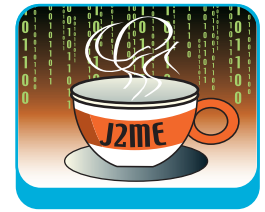



Fig.3: *Form per aggiungere, visualizzare ed aggiornare un elemento video*

composto da un array di byte identificato, all'interno del RecordStore, da un ID numerico positivo. Quest'ultimo può anche essere equiparato ad una primary-key o al ROWID utilizzato da Oracle per identificare ogni singolo record all'interno di una tabella. L'invocazione del seguente metodo gestisce le operazioni di aggiunta e aggiornamento del VideoItem passato come argomento. In base al valore della variabile intera id, contenuta nell'oggetto VideoItem, il metodo stabilisce se eseguire un nuovo inserimento oppure l'aggiornamento di un elemento esistente.

```
public void saveVideo(VideoItem video) {
    ByteArrayOutputStream baos = new
                    ByteArrayOutputStream();
    DataOutputStream dos = new
                    DataOutputStream(baos);
    this.write(video.getTitle(), dos);
    this.write(video.getAuthor(), dos);
    this.write(""+video.getYear(), dos);
    this.write(video.getNotes(), dos);
    byte[] record = baos.toByteArray();
    if (video.getId() <= 0) { // id <= 0, aggiunge
                    un nuovo record ...
        try {
            this.videoStore.addRecord(record, 0,
                    record.length);
        } catch (RecordStoreException ex) {
            ex.printStackTrace(); }
    } else { // ... record esistente: azione di update
        try {
            this.videoStore.setRecord(video.getId(),
                    record, 0, record.length);
        } catch (RecordStoreException ex) {
            ex.printStackTrace();}
    } }
  /* scrive la stringa di testo sullo stream di output
                    specificato */
  private void write(String text, DataOutputStream
                    dos) {
    text = text == null ? "" : text;
    try {
        dos.writeUTF(text);
    } catch (IOException ex) {
        ex.printStackTrace();}
  }
```

Per semplificare la fase di scrittura dell'oggetto VideoItem, è stato utilizzato un oggetto di tipo DataOutputStream che fa da wrapper ad un ByteArrayOutputStream. Analogamente, il processo di lettura di un VideoItem sarà gestito attraverso un ByteArrayInputStream contenuto all'interno di un DataInputStream.

```
private VideoItem getVideoById(int id) {
    VideoItem video = new VideoItem();
    // crea un DataInputStream partendo dall'array
                    di byte ...
    DataInputStream dis = null;
    try {
        dis = new DataInputStream(new
                    ByteArrayInputStream(
                    this.videoStore.getRecord(id)));
    } catch (RecordStoreException ex) {
        ex.printStackTrace();}
    // ... e legge le proprietà del video
    video.setId(id);
    video.setTitle(this.read(dis));
    video.setAuthor(this.read(dis));
    video.setYear(Integer.parseInt(this.read(dis)));
    video.setNotes(this.read(dis));
    return video;}
  /* legge la stringa in formato UTF dallo stream
                    specificato */
  private String read(DataInputStream dis) {
    String value = null;
    try {
        value = dis.readUTF();
    } catch (IOException ex) {
        return null;}
    value = value.equals("") ? null : value;
    return value;
  }
```

Così come la fase di lettura, anche quella di eliminazione si basa sulla conoscenza dell'identificativo del record su cui si vuole andare ad operare. Il seguente stralcio di codice illustra come rimuovere un record dal RecordStore.

```
public void deleteVideo(int videoId) {
    try {
        this.videoStore.deleteRecord(videoId);
    } catch (InvalidRecordIDException ex) {
        System.err.println("Identificativo " + videoId
                    + " non valido!");
    } catch (RecordStoreNotOpenException ex) {
        System.err.println("VideoStore non aperto");
    } catch (RecordStoreException ex) {
        System.err.println("Errore durante
                    l'eliminazione del record"); }
  }
```

L'identificativo del record rimosso non verrà più utilizzato durante il rimanente ciclo di vita del RecordStore.

SORT & SEARCH

Realizzeremo due nuove features: visualizzazione dell'intero catalogo video e ricerca di uno o più elementi video in base ad una chiave immessa dall'utente. Entrambe le funzionalità dovranno presenta-

re i risultati ordinati in base al titolo del video. Per semplificare la fase di ordinamento basterà implementare l'interfaccia *RecordComparator.*

```
class VideoItemComparator implements
                                        RecordComparator {
    /* stream di lettura */
    private DataInputStream dis;
    /** Interface method implementation */
    public int compare(byte[] b, byte[] b0) {
        // legge i titoli
        int val = this.getTitle(b).compareTo(
                                        this.getTitle(b0));
        if (val < 0) {
            val = RecordComparator.PRECEDES;
        } else if (val > 0) {
            val = RecordComparator.FOLLOWS;
        } else {
            val = RecordComparator.EQUIVALENT;}
        return val; }
    /* legge il titolo del video descritto dallo
                                specifico array di byte */
    private String getTitle(byte[] rec) {
        dis = new DataInputStream(new
                                ByteArrayInputStream(rec));
        String title = null;
        try {
            title = dis.readUTF().toLowerCase();
        } catch (IOException ex) {
            ex.printStackTrace();}
        return title; }
}
```

Il "cuore" di questa classe è rappresentato dal metodo *compare* che riceve come parametri due byte array, ognuno dei quali rappresenta un record. Il compito di questo metodo è quello di determinare l'ordine di precedenza tra i due record specificati. Il valore intero di ritorno deve essere una delle costanti esposte dalla stessa interfaccia: *EQUIVALENT, PRECEDES* e *FOLLOWS.* Anche il criterio di ricerca è realizzabile attraverso l'uso di un'altra interfaccia esposta dal sistema RMS: *RecordFilter.*

```
class VideoItemFilter implements RecordFilter {
    /* chiave di ricerca */
    private String matcher;
    /* stream di lettura */
    private DataInputStream dis;
    /**
     * Crea un nuovo filtro con una determinata
                                        chiave di ricerca
     *
     * @param matcher chiave di ricerca da
                                        associare al filtro
     */
    public VideoItemFilter(String matcher) {
        this.matcher = matcher.toLowerCase();}
```

```
    /* Interface method implementation */
    public boolean matches(byte[] b) {
        dis = new DataInputStream(new
                                ByteArrayInputStream(b));
        String title = null;
        try {
            title = dis.readUTF().toLowerCase();
        } catch (IOException ex) {
            ex.printStackTrace();
            return false; }
        return title.indexOf(matcher) >= 0;}
}
```

Il metodo *matches* riceve un array di byte rappresentante il contenuto di un record e ritorna true se la chiave di ricerca inserita è un subset (case insensitive) del titolo del record stesso. A questo punto non ci rimane che utilizzare le classi appena create invocando il metodo *enumerateRecords* di *RecordStore* di cui riportiamo la dichiarazione:

```
public RecordEnumeration enumerateRecords(
        RecordFilter filter, RecordComparator comparator,
                                        boolean keepUpdated)
throws RecordStoreNotOpenException
```

Il valore di ritorno rappresenta una struttura, bidirezionalmente navigabile, che gestisce una sequenza logica di record in base ai parametri specificati. Potremmo quindi implementare la funzione di ordinamento dell'intero catalogo video invocando il metodo nel seguente modo:

```
RecordEnumeration en =
                videoStore.enumerateRecords(null,
                new VideoItemComparator(), false));
```

mentre per la fase di ricerca si dovrà aggiungere l'impementazione del *RecordFilter* costruita sulla chiave di ricerca inserita dall'utente:

```
RecordEnumeration en = videoStore.enumerateRecords(
                        new VideoItemFilter(key), new
                        VideoItemComparator(), false));
```

CONCLUSIONI

In questo articolo abbiamo visto come, attraverso le API fornite dal sistema RMS, sia possibile memorizzare dati in maniera non volatile su un dispositivo mobile. Questa soluzione rappresenta la scelta più adeguata e portabile quando si vuole dare all'utente la possibilità di memorizzare e recuperare dati inerenti la logica dell'applicazione, come nell'esempio descritto, o quando si vogliono rendere persistenti gli stati di configurazione della *MIDlet suite.*

Fabrizio Fortino

RECORDSTORE CONDIVISI

Una delle caratteristiche più interessanti introdotte con la versione 2.0 del *MID Profile* è sicuramente la possibilità di condividere i *RecordStore* con le altre *MIDlet suite* installate sul dispositivo. Questa funzionalità risulta essere molto importante quando si sviluppano *MIDlet suite* differenti a cui si vuole dare la possibilità di "comunicare" tra di loro. Per far sì che un determinato *RecordStore* sia condivisibile è necessario che la *MIDlet suite,* proprietaria della base dati, imposti la modalità di accesso (attraverso il metodo *setMode*) ad *AUTHMODE_ANY.* Un'altra applicazione che vorrà accedere a questa base dati condivisa dovrà dichiararlo esplicitamente utilizzando un'overload del metodo *openRecordStore* con i seguenti parametri: nome del *RecordStore,* nome del *vendor* e nome della *MIDlet suite* associata alla base dati.

JMX CONTROLLA I TUOI PROGRAMMI

ILLUSTREREMO UNA TECNICA CHE CONSENTE A UN QUALUNQUE CLIENT DI INTERROGARE UN SOFTWARE IN ESECUZIONE ALLO SCOPO DI CONOSCERNE LO STATO DI FUNZIONAMENTO O ALTRI PARAMETRI

Che cosa è JMX? In un primo momento la risposta che daremo sembrerà decisamente "astratta". Vi proponiamo due definizioni: la prima ricavata da wikipedia: *"JMX: Java Management Extension è un'insieme di specifiche, pattern che permettono di inserire all'interno di una applicazione sviluppata in "java dei componenti per il monitoraggio della stessa, chiamate Mbean".* Una seconda ricavata dal sito di Sun *"JMX (Java Management eXtensions), fornisce un metodo semplice e standard per gestire risorse applicativi dispositivi".* Cerchiamo dunque di chiarirci le idee, tentando di associare a queste due definizioni abbastanza "fumose" un esempio pratico che ci consenta di traslare da un piano strettamente teorico a un piano decisamene più pratico. L'idea è la seguente: abbiamo costruito un'applicazione Web al cui interno esiste un modulo per la registrazione degli utenti. Ma quanti utenti abbiamo registrato? quanti utenti hanno tentato una registrazione e poi hanno abbandonato prima della fine? Quali referrer hanno portato gli utenti a registrarsi sul nostro sito? Se volessimo implementare una soluzione per dare risposta a questo tipo di problemi, potremmo semplicemente creare delle interfacce specifiche che una nostra applicazione potrebbe interrogare, per avere delle risposte immediate. È esattamente quello che faremo tramite JMX. Perché JMX si differenzia dunque rispetto a questo tipo di soluzione? Per il semplice motivo che JMX è un insieme di specifiche per il disegno di interfacce che consentano di monitorare/gestire il comportamento di un'applicazione. Saremo noi a stabilire quali sono le risorse da monitorare/gestire "incastrandole" in un contenitore che chiameremo *Mbean.* Ogni Mbean dovrà rispettare delle specifiche particolari per essere compatibile con JMX. Al solito il vantaggio della tecnologia è quella di essere uno standard, per cui siamo sicuri che qualunque applicazione che utilizzi JMX sarà in grado di essere interrogata tramite un client che utilizzi il medesimo insieme di specifiche. Nel set di librerie di Java 1.5 è compresa anche *"jconsole",* un applicazione che mediante una GUI essenziale ma di immediato utilizzo permette di collegarsi e gestire qualsiasi applicazione JMX. Dopo questa introduzione, necessariamente lunga, andiamo ad iniziare.

ARCHITETTURA DI JMX

Come già detto al centro di JMX ci sono gli *MBean*, classi Java che rispettano gli standard JavaBean per il naming. Ad esempio una proprietà *"xxx"* di tipo int in lettura e scrittura dovrà essere implementata secondo i due metodi:

```
void setXxx(int val){
        [...] }
int getXxx(){
        [...] }
```

Volendo invece pubblicare la proprietà *"yyy"* di tipo *String* in sola lettura il bean dovrà implementare il solo metodo

```
String setYyy(){...}
```

Gli *MBean* hanno il compito di esporre le proprietà e i metodi che hanno una rilevanza per l'applicazione dal punto di vista della sua gestione. In sostanza all'interno del nostro *MBean* dovremo implementare tutte quelle strutture che vogliamo monitorare o gestire. Un *MBean* è definito da un'interfaccia e da una classe che la implementa. I metodi e le proprietà esposti sono quelli definiti nell'interfaccia.
Il nome dell'interfaccia deve obbligatoriamente ter-

Fig. 1: **La finestra di connessione di "jconsole" nella quale digitare l'indirizzo al quale risponde l'MBean Server**

minare per *"MBean"*. È necessario poi scrivere l'implementazione di tale interfaccia. In questo caso non ci sono vincoli sul naming. Un'istanza del bean deve poi essere registrata presso un *"MBean server"* assegnandogli contestualmente un nome univoco. Il client JMX si collegherà proprio a tale *MBean* server per ottenere un riferimento agli *MBean* ivi pubblicati.

L'APPLICAZIONE DI ESEMPIO

Come già detto, la nostra applicazione dovrà semplicemente gestire un gruppo di utenti registrati, da un lato non consentendo la registrazione oltre un certo numero, dall'altro implementando un metodo per ricercare un utente tramite il suo username. Una Map memorizzerà tutti gli utenti registrati usando come chiave l'username stesso. In questo modo la ricerca di username eventualmente già utilizzati e la ricerca di un utente a partire dall'username sarà molto rapida. La classe implementa l'interfacia *Runnable,* in modo da poter essere attivata in un thread separato. Come noto in questi casi il metodo *run(),* specificato nell'interfaccia *Runnable,* viene eseguito quando attraverso la classe *Thread* si crea un nuovo flusso di esecuzione. Il primo e l'ultimo metodo invocati da *run()* saranno *startup()* e *shutdown(),* richiamati rispettivamente nel momento in cui la nostra applicazione server per la gestione degli utenti parte e nel momento in cui viene disattivata. Il ciclo principale nel metodo *run()* fa si che l'applicazione non termini sino a quando l'attributo running non sia valorizzato a *false.* Degli altri metodi riportiamo solo la signature: *maxUserAllowed()* fornisce il massimo numero di utenti gestibili dal sistema, *getUpTime()* la data in cui il sistema è stato attivato, i metodi *addUser(), removeUser()* e *findUserByUsername()* permettono rispettivamente di aggiungere un utente, rimuoverne uno registrato e trovarne uno dallo username.

```
package ioprogrammo.server;
public final class Server implements Runnable {
    private int maxUserAllowed;
    /** while true the server is up and running */
    private boolean running = true;
    private final Date bootTime = new Date();
    private Map<String, User> users;
    private Server() {
        reset();}
    public void reset() {
        users = new HashMap<String, User>();
        maxUserAllowed =
                    DEFAULT_MAX_USER_ALLOWED;}
    public int getMaxUserAllowed() {[...]}
    public void setMaxUserAllowed(int
                          maxUserAllowed) {[...]}
    public int getNumberOfRegisteredUser() {
                          return users.size();}
    public Date getUpTime() {return bootTime;}
    public void addUser(String username, String
            password) throws ServerException {[...]}
    public void removeUser(User user) {[...]}
    public User findUserByUsername(String username)
                                          {[...]}
    public void run() {
        startup();
        while (running) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace(); } }
        shutdown(); }
    private void startup() {[...]}
    private void shutdown() {[...]}
    public void halt() {
        running = false; }
    public static void main(String[] a) {
        new Thread(new Server()).start(); }
}
```

INTERFACCIA DEL MBEAN

Il primo *MBean* che realizzeremo permetterà di offrire dati statistici sugli utenti registrati e sull'applicazione in generale. Si desidera conoscere il numero massimo di utenti ammessi, il numero di utenti effettivi, la data in cui il sistema è stato avviato, da quanto tempo il server è attivo e una media che dia il numero di utenti registrati per ora. Definiamo quindi un'interfaccia dal nome *StatisticsMBean* con gli opportuni metodi. Ricordate che JMX impone che ogni *MBean* implementi un'interfaccia il cui nome termini per *"MBean"*. I metodi e le coppie setter/getter definiti nell'interfaccia saranno gli unici accessibili da remoto. C'è anche da ricordare che nell'interfaccia non è possibile utilizzare qualsiasi tipo di dato come parametro o tipo di dato restituito ma è meglio limitarsi ai tipi primitivi e ai rispettivi wrapper. Nei prossimi articoli avremo modo di approfondire questa questione. Per ora popoliamo la nostra interfaccia con soli getter che forniscono le

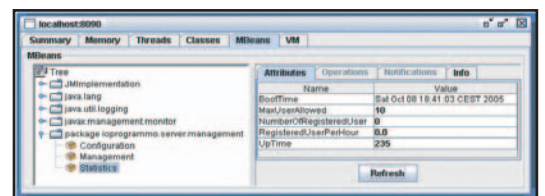



Fig. 2: ***Ecco la schermata che dovrebbe presentare jconsole per la visualizzazione del MBean che pubblica dati statistici sull'applicazione***

informazioni desiderate.

```
package ioprogrammo.server.management;
import java.util.Date;
public interface StatisticsMBean {
    public int getMaxUserAllowed();
    public int getNumberOfRegisteredUser();
    public Date getBootTime();
    public long getUpTime();
    public double getRegisteredUserPerHour();
}
```

IMPLEMENTAZIONE DEL MBEAN

JMX prevede che ogni interfaccia *MBean*, abbia un'implementazione. La strada più semplice sarebbe quella di scrivere semplicemente un'implementazione dei metodi sopra definiti. Per comodità, invece, definiamo prima di tutto una classe *SuperMBean* che subclasseremo per ogni MBean di cui avremo necessità. Seguiamo questa strategia perché come vedremo tutti gli *MBean* che utilizzeremo avranno dei metodi in comune. Ad esempio, come spiegato nell'introduzione, ad ogni *MBean* deve essere assegnata una stringa identificativa univoca contestualmente alla sua registrazione presso l'*MBean Server*. È corretto ed in puro approccio OOP assegnare la responsabilità di conoscere la propria stringa identificativa al *MBean* stesso. La stringa viene costruita concatenando il nome del package e della classe del bean. Oltre a ciò tutti i nostri *MBean* dovranno tenere sotto controllo un'istanza di Server. Anche in questo caso è più elegante settare il puntatore all'istanza di Server da gestire in questa super classe, di cui è riportato schematicamente il codice. Il costruttore accetta un'istanza di Server, che sarà l'oggetto gestito. Il nome dell'*MBean* viene generato dal metodo *getObjectNameString()*. In JMX il nome dell'oggetto non è una semplice stringa ma un oggetto di tipo *ObjectName*.

```
public class MBean {
    private ObjectName myName;
    private Server managed;
    public MBean(Server managed) {
        String objectName = null;
        try {
            objectName = getObjectNameString();
            myName = new ObjectName(objectName);
        } catch (MalformedObjectNameException e) {
            throw new RuntimeException(objectName
                    + " is a malformed ObjectName", e);}
        this.managed = managed;}
    public ObjectName getObjectName() {return
myName;}
    public Server getManaged() {return managed;}
    public String getObjectNameString() {
        String domain =
                this.getClass().getPackage().toString();
        String name =
                    this.getClass().getSimpleName();
        return domain + ":type=" + name; }
}
```

Definiamo a questo punto l'implementazione vera e propria del MBean che ovviamente implementerà l'interfaccia *StatisticheMBean* ed estenderà la classe *MBean*. L'implementazione si limita ad invocare i corretti metodi sul server per restituire i dati della statistica.
L'implementazione è abbastanza semplice.

```
package ioprogrammo.server.management;
import ioprogrammo.server.Server;
import java.util.Date;
public class Statistics extends MBean implements
                                        StatisticsMBean {
    public Statistics(Server managed) {
        super(managed); }
    public int getMaxUserAllowed() {
        return getManaged().getMaxUserAllowed(); }
    public int getNumberOfRegisteredUser() {
        return
         getManaged().getNumberOfRegisteredUser();}
    public void shutdown() {
        getManaged().halt(); }
    public Date getBootTime() {
        return getManaged().getUpTime();}
    public long getUpTime() {
        return (int) (.001 * (new Date().getTime() -
                    getBootTime().getTime())); }
    public double getRegisteredUserPerHour() {
        double hour = (double) getUpTime() / 3600;
        return getManaged()
                .getNumberOfRegisteredUser() / hour; }
}
```

REGISTRAZIONE DEL MBEAN NEL MBEAN SERVER

La procedura da eseguire per registrare un *MBean* in un *MBean* server è abbastanza lunga e assolutamente ripetitiva. Si deve eseguire il lookup del server, creare l'*ObjectName* da associare al nuovo *MBean*, registrarlo e gestire eventuali eccezioni. Non è quindi pensabile ripetere tutta la procedura ogni volta che sia necessario registrare un *MBean*. Per non appesantire il codice di chiamate ripetute definiamo una classe di supporto che si occupa della registrazione degli *MBean*, attraverso il metodo

registerMBean(). Il metodo chiede al *MBean* stesso la sua stringa di registrazione. Nel codice, a scopo illustrativo, sono riportate tutte le eccezioni che possono essere sollevate. Probabilmente si potrebbe optare per un più conveniente *catch(Exception e)*.
La classe a cui richiedere la registrazione degli *MBean* è *MbeanServer* che può essere ottenuta dalla classe *ManagementFactory* attraverso il metodo *getPlatformMBeanServer()* che restituisce un puntatore al *MbeanServer* di default per l'istanza della JVM su cui sta girando l'applicazione. Come è possibile notare la registrazione del bean non presenta eccessive difficoltà. È sufficiente invocare il metodo *registerMBean()* passando l'implementazione del bean e l'*ObjectName* da associargli. Le eccezioni sollevate riguardano il fatto di aver utilizzato un *ObjectName* già utilizzato da un diverso *MBean* già registrato, un generico problema durante la registrazione ed infine il tentativo di registrare un *MBean* che non rispetti le specifiche JMX, ad esempio perché permette di impostare delle proprietà di un tipo non supportato o perché non rispetta i vincoli sui nomi delle interfacce.

```
package ioprogrammo.server.management;
public class MBeanHelper {
    private static final MBeanServer mBeanServer =
        ManagementFactory.getPlatformMBeanServer();
    private MBeanServer getMBeanServer() {
        return mBeanServer; }
    public void registerMBean(MBean mBean){
        MBeanServer mbs = getMBeanServer();
        try {
            mbs.registerMBean(mBean,
                                  mBean.getObjectName());
        } catch (InstanceAlreadyExistsException e) {
            e.printStackTrace();
        } catch (MBeanRegistrationException e) {
            e.printStackTrace();
        } catch (NotCompliantMBeanException e) {
            e.printStackTrace(); } }
}
```

Ora dobbiamo modificare il server affinchè in fase di startup registri l'*MBean* presso il *MbeanServer.* Per fare ciò ci vengono in aiuto i metodi *startup()* e *shutdown()* che tanto saggiamente avevamo previsto nella stesura del server. Nel metodo startup quindi creeremo un *Mbean* di tipo *StatisticsMBean* e ne delegheremo la registrazione alla classe di supporto appena creata. Scriviamo quindi il corpo del metodo startup.

```
private void startup() {
    System.out.println("starting up");
    MBeanHelper.instance().registerMBean(
                                 new Statistics(this));
}
```

COLLEGARSI CON JCONSOLE

Aprite la directory dove è installato il Java *Development Kit* (attenzione non il JRE), portatevi nella cartella *"bin"* e lanciate *"jconsole"*. Nella schermata iniziale scegliete la linguetta *"remote"*, come *"host"* immettete *"localhost"*, come *"port"* lo stesso numero che avete specificato in corrispondenza del parametro *com.sun.management.jmxremote.port,* in questo caso 8090 e cliccate su *connect.* Se tutto procede correttamente apparirà una finestra che mostrerà nella sezione sinistra tutti gli *Mbeanregistrati* suddivisi in categorie. Le prime categorie raggruppano tutti gli *MBean* della Java virtual machine. Spiegare i significati di tutti gli *MBean* pubblicati dalla *Java Virtual Machine* è fuori dagli scopi dell'articolo, tuttavia alcuni di essi sono autoesplicativi.
L'ultima categoria è *"package ioprogrammo.server.management".* Aprendo tale categoria scegliete il bean *"Statistics"* e vedrete elencati tutti i parametri esposti dall'*MBean.* Facendo doppio click su uno dei parametri di tipo intero è anche possibile vedere un evoluzione temporale del parametro stesso.
Ecco quindi che lo stato dell'applicazione è visibile e consultabile via web tramite JMX.

LANCIARE IL SERVER

Dopo aver compilato l'applicativo eseguitelo avendo cura di impostare i parametri specificati. Il parametro *"port"* identifica la porta TCP su cui si deve mettere in ascolto il *Mbean* Server. Potete scegliere una qualunque porta libera.
I parametri *"authenticate"* e *"ssl"* a false disabilitano qualsiasi impostazione di sicurezza, in modo che questo primo esempio possa girare senza problemi.
Nei successivi articoli approfondiremo anche tale aspetto.

```
java -Dcom.sun.management
  .jmxremote -Dcom.sun.management
    .jmxremote.port=8090 -Dcom.sun
 .management.jmxremote.authenticate
       =false -Dcom.sun.management
               .jmxremote.ssl= false
                 ioprogrammo.Server
```

CONCLUSIONI

I concetti che stanno alla base di JMX non sono complessi e l'utilità è indiscutibile. Nel nostro esempio abbiamo complicato leggermente le cose creando una superclasse da cui derivare e una classe per la registrazione, tuttavia questa metodologia vi consentirà di risparmiare molto tempo nel momento in cui vorrete utilizzare questa tecnologia in fase di produzione.
Ci sono ancora molte cose da dire rispetto a JMX. Sicuramente torneremo sull'argomento nei prossimi numeri.

Daniele De Michelis

INSTALLAZIONI IN UN CLICK(ONCE)

UTILIZZIAMO UNA DELLE ULTIME TECNOLOGIE DI CASA MICROSOFT PER DISTRIBUIRE IN MODO EFFICACE LE NOSTRE APPLICAZIONI SENZA DOVERCI PREOCCUPARE DI FUTURI AGGIORNAMENTI E DELL'OMOGENEITÀ DELLE VERSIONI INSTALLATE

Quali sono i problemi legati all'installazione del software? È facile fornire qualche riposta immediata:

1) Difficoltà nella distribuzione degli aggiornamenti.
2) Difficoltà nel deployment delle applicazioni.

Sembrerebbero due problemi di facile soluzione. Tuttavia ci accorgeremo che proprio la fase di installazione della versione finale e delle versioni intermedie di un'applicazione rappresenta da solo uno degli scogli più importanti per uno sviluppatore. Nonostante la distribuzione di un'applicazione sia il passo finale del ciclo di produzione, non deve essere considerata ultimo anche in ordine di importanza. Stabilire a priori la modalità di distribuzione e installazione del software è fondamentale per una buona riuscita del prodotto. Spesso le applicazioni devo essere immediatamente fruibili da parte dell'utente finale, facilità di installazione e manutenzione costituiscono requisiti importantissimi. Questo tipo di problematica ha favorito nel corso del tempo lo sviluppo di applicazioni Web che per loro natura non necessitano di installazione e vengono aggiornate semplicemente lato server. D'altro canto è noto che lo sviluppo di applicazioni Web non ha per certi versi la stessa flessibilità di un'applicazione standalone. Da qui la necessità di un modello capace di rendere semplice e sicura la distribuzione delle applicazioni client, su un piano simile alla distribuzione delle applicazioni Web. *ClickOnce* consente di ottenere esattamente questo: una distribuzione semplice, efficace, sicura delle applicazioni Windows, non invasiva sul client di destinazione. Ovviamente non tutte le applicazioni possono usufruire di questo modello. Proveremo ora ad analizzare le caratteristiche di *ClickOnce* per valutare in che modo può esserci utile nel lavoro di tutti i giorni. Immaginiamo di dover sviluppare un'applicazione per un nostro ipotetico cliente, la cui necessità sia quella di implementare un servizio di registrazione anagrafiche distribuito, rilasciato a tutte le sue agenzie geograficamente distanti tra loro. Ogni agenzia utilizza il software *myAnag* per registrare quotidianamente i dati dei nuovi clienti acquisiti. Va da sé che tutte le agenzie devono possedere in ogni determinato istante la stessa identica versione del software *myAnag*.

PRIMA DISTRIBUZIONE CON CLICKONCE

Terminata la fase di progettazione, sviluppo e test della nostra applicazione, siamo pronti a procedere con la distribuzione su tutti i client. Apriamo la nostra soluzione in Visual Studio 2005 ed accediamo al modello di distribuzione *ClickOnce* attraverso il menu *Build/Compila | Publish nomeApp*, dove *nomeApp* rappresenta il nome dell'applicazione che stiamo distribuendo. Visual Studio 2005 presenta un wizard che permette la selezione della modalità di distribuzione che desideriamo applicare. La prima scheda del wizard consente di scegliere tra quattro diverse modalità: una cartella sul disco, una condivisione di rete, un server FTP o un sito web. In realtà è anche possibile distribuire l'applicazione utilizzando supporti come CD, DVD

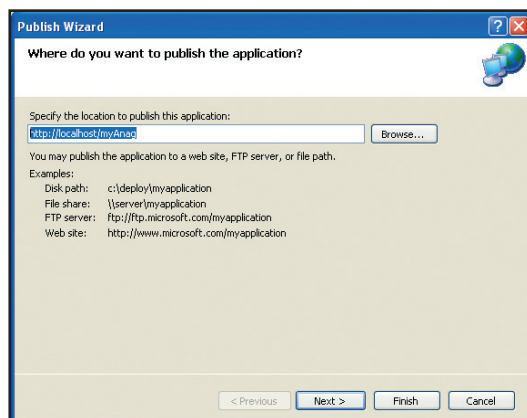



Fig. 1: **Impostiamo il percorso di pubblicazione**

o Pen Drive. Data la differente posizione geografica dei client, scegliamo di distribuire *myAnag* attraverso un server web, specificando, appunto, il percorso di pubblicazione in forma di URL. La seconda scheda del wizard chiede di impostare la modalità di esecuzione dell'applicazione sul client. L'applicazione può, infatti, essere resa disponibile in due modi: online e offline, oppure solo online. Nella prima ipotesi viene creato un collegamento dal menu *start* di Windows e l'applicazione può anche essere disinstallata, con il vantaggio di poter eseguire l'applicazione anche senza essere connessi al sito web di pubblicazione. Nella seconda ipotesi i client devono, invece, connettersi ogni volta al sito di pubblicazione per poter eseguire l'applicazione.

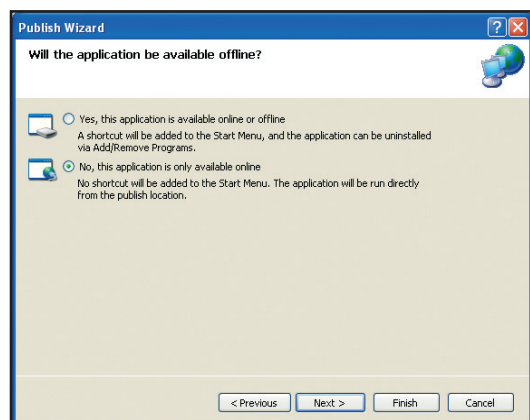



***Fig. 2:* Impostiamo la modalità di installazione**

Selezioniamo la seconda ipotesi e terminiamo il wizard. Visual Studio compila il progetto, crea i file di pubblicazione ed infine copia i file in una cartella all'interno della directory da noi specificata. Viene creata una cartella, il cui nome è composto dal nome dell'applicazione più il numero di versione pubblicato, contenente tutti i file associati al deploy corrente. Al termine della procedura di pubblicazione, Visual Studio 2005 mostra la pagina Web generata dove sostanzialmente vengono riportati il nome dell'applicazione, l'ultima versione pubblicata e, se disponibile, l'eventuale publisher. Utilizzando l'URL di pubblicazione, quindi, i client possono eseguire l'applicazione cliccando sul tasto *Run*. A questo punto il runtime *ClickOnce* esegue i controlli di sicurezza per verificare innanzitutto se l'applicazione è firmata e se la firma è valida, e poi, tramite *Code Access Security*, se i permessi richiesti sono compatibili con quelli impostati per l'esecuzione o l'installazione dalla *Internet Zone*, che è la zona di provenienza/pubblicazione che stiamo utilizzando. Qualora almeno uno di questi controlli fallisse, ci verrà mostrato un *Security Warning*, una window che chiede all'utente una conferma sull'azione da eseguire, cioè se l'applicazione deve essere eseguita o meno. Confermiamo l'esecuzione cliccando su *Run*. Come possiamo vedere l'applicazione viene correttamente eseguita.

***Fig. 3:* La security warning**

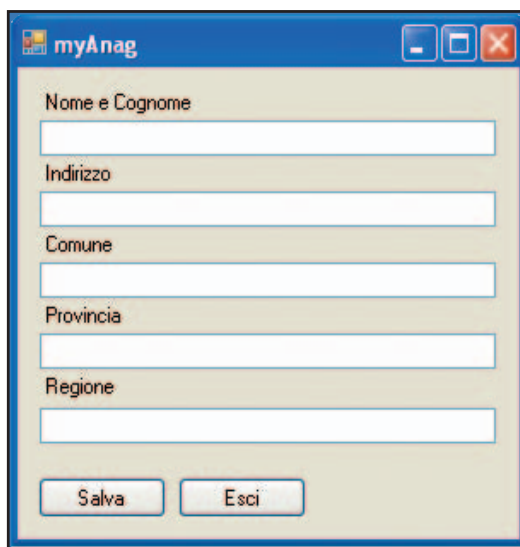



***Fig. 4:* L'applicazione in esecuzione sul client**

I TUOI APPUNTI

La *Security Warning* può essere evitata e vedremo come fare firmando l'applicazione *ClickOnce* e impostando i permessi per una corretta esecuzione.

ESECUZIONE DELL'APPLICAZIONE IN MODALITÀ OFFLINE

Supponiamo che ora il nostro cliente desideri rendere l'applicazione utilizzabile anche da

DISINSTALLAZIONE DELLE APPLICAZIONI CLICKONCE

La modalità di installazione offline, oltre ad aggiungere un collegamento dal menu *Start* di Windows, consente anche di rimuovere la versione installata dell'applicazione *ClickOnce* ripristinando la versione precedente, utilizzando il tool di Windows *Add/Remove Programs*, accessibile dal *Pannello di Controllo*.
La stessa procedura può essere utilizzata per rimuovere definitivamente l'applicazione dal sistema client.

quei client che non sono sempre connessi alla rete, modificando appunto un requisito fondamentale della prima distribuzione. Nessun problema. Riapriamo la soluzione in Visual Studio 2005 e rieseguiamo la distribuzione sempre utilizzando il menu *Build | Publish myAnag*. Nel wizard lasciamo invariato il percorso di pubblicazione e modifichiamo, invece, l'opzione di disponibilità dell'applicazione impostando la modalità online e offline, come visualizzato in **Figura 5**, e terminiamo il wizard.

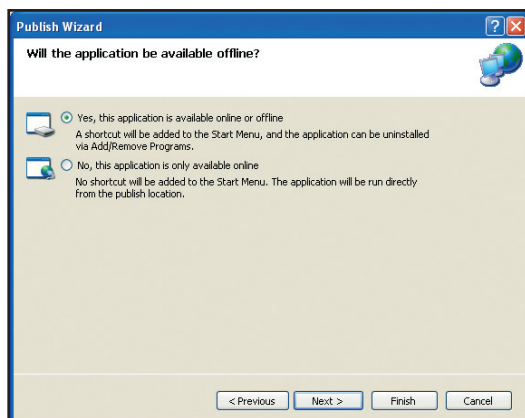



Fig. 5: Consentiamo l'esecuzione anche offline

Visual Studio 2005 riesegue la compilazione del progetto, incrementando il numero di build, e riesegue la pubblicazione del progetto nella cartella da noi indicata. Ora la pagina di publish presenta ancora una volta il nome dell'applicazione, il numero di versione, questa volta incrementato di build, e non più il tasto *Run*, ma il tasto Install. Cliccando su install *ClickOnce* riesegue i controlli di sicurezza, visualizzando la *Security Warning* qualora non validi, provvede ad eseguire una installazione dell'applicazione sul client e infine esegue l'applicazione appena installata. La differenza con la precedente modalità di distribuzione è data dalla presenza di un collegamento, nel menu *Start | Programmi di Windows*, che consente di eseguire l'applicazione myAnag anche se non si è connessi alla rete o comunque al percorso di pubblicazione. L'installazione sul client non è reale o comunque non simile a quanto avviene con il modello di setup classico. L'applicazione viene in pratica installata localmente nella cache di *ClickOnce* ed eseguita da quella posizione dopo aver verificato che nella cartella di pubblicazione, se disponibile, non è presente una nuova versione, nel qual caso viene chiesto se effettuare l'aggiornamento.

I FILE MANIFEST DI CLICKONCE: .DEPLOY E .APPLICATION

La distribuzione e l'esecuzione delle applicazioni *ClickOnce* sono basate su due fondamentali file *manifest:*

.deploy	descrive le informazioni di distribuzione su cui spiccano la descrizione della versione attualmente disponibile, il riferimento all'application manifest, le modalità di installazione e di update dell'applicazione e la firma del *manifest* di deploy;
.application	descrive nel maggior dettaglio l'applicazione distribuita con *ClickOnce*, riportando le informazioni di identificazione sull'assembly principale distribuito e su tutte le sue relative dipendenze.

L'esecuzione dell'applicazione è associata al file *.application* il cui percorso è passato come parametro alla libreria *dfshim.dll* eseguita utilizzando il comando *rundll.exe*.

UPDATE DELL'APPLICAZIONE

Supponiamo che venga richiesto di sostituire, nel nostro form di inserimento anagrafiche, il campo di testo per l'inserimento delle regioni con una combo box che, in maniera più semplice, ne consente la selezione. Dopo aver aggiornato l'applicazione ripetiamo il processo di pubblicazione visto nel precedente paragrafo. Al termine del processo di generazione della nuova build e della successiva pubblicazione da parte del wizard, Visual Studio 2005 mostra ancora una volta la pagina di pubblicazione. Ora, per verificare che l'update dell'applicazione funziona correttamente, proviamo ad eseguire l'applicazione dal menu *start* di Windows, proprio come farebbe un ipotetico utente. Il risultato è quello che possiamo vedere in **Figura 6**.

Fig. 6: Il prompt che avvisa della disponibilità di una nuova versione

ClickOnce accerta la presenza di una nuova versione e chiede all'utente se desidera effettuare l'aggiornamento. Come avviene il processo? Internamente viene generato un hash per ogni build dell'applicazione. Se la cartella di pubblicazione è disponibile, *ClickOnce* verifica l'hash installato con l'hash della versione più recente disponibile. Se quest'ultimo è maggiore a quello installato, *ClickOnce* avvisa l'utente della disponibilità di una nuova versione e chiede se eseguire l'aggiornamento, altrimenti procede con la normale esecuzione dell'applicazione. Al ter-

mine dell'aggiornamento, *ClickOnce* esegue la nuova versione dell'applicazione installata.

AGGIORNARE L'APPLICAZIONE PROGRAMMATICAMENTE

Questo approccio va bene per gli aggiornamenti poco frequenti. Ma se si prevedono aggiornamenti frequenti, anche durante l'utilizzo dell'applicazione, come possiamo notificare ai client la presenza di nuovi aggiornamenti sul server? *ClickOnce* si avvale di un set di API disponibili attraverso il *namespace System.Deployment*. Tale *namespace* consente, infatti, di verificare la disponibilità di nuove versioni ed eventualmente di procedere al download. Predisponiamo quindi un box dove un *alert* ci avvisa della presenza di aggiornamenti, ed un *button* che consente di aggiornare l'applicazione. Aggiungiamo al form un controllo *GroupBox* ed inseriamo al suo interno un controllo label ed un button. Aggiungiamo poi al form un controllo *timer* che ad intervalli regolari verifica la presenza di aggiornamenti.

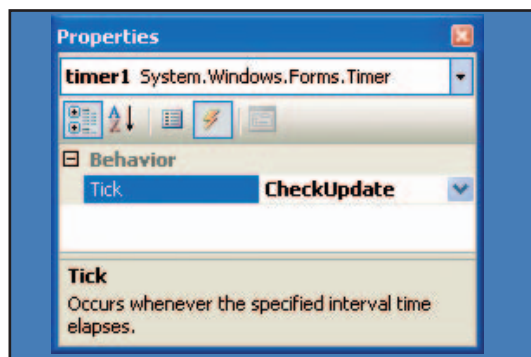



Fig. 7: ***La property box e la gestione dell'evento Tick***

Impostiamo il timer su un intervallo di 30 secondi modificando la proprietà Interval su 30000, in millisecondi. Selezioniamo il metodo da richiamare allo scadere dell'intervallo impostando il valore dell'evento *Tick* su *CheckUpdate* e premiamo invio sulla propertybox per generare il relativo codice.
Nel codice del form importiamo il namespace *System.Deployment.Application*:

```
using System.Deployment.Application;
```

nel metodo *CheckUpdate* inseriamo il seguente codice:

```
ApplicationDeployment deploy =
ApplicationDeployment.CurrentDeployment;
    if (deploy.CheckForUpdate())
```

```
    {
        this.lblUpdate.Text = "È disponibile una
                                    nuova versione.";
        this.btnUpdate.Enabled = true;
    }
    else
    {
        this.lblUpdate.Text = "Nessun
                              aggiornamento disponibile.";
        this.btnUpdate.Enabled = false;
    }
```

LA CODE ACCESS SECURITY

La *CAS* è presente fin dalla prima versione del Microsoft .NET Framework e consente di prevenire l'esecuzione di codice non attendibile o che richiede elevati permessi per essere eseguito. Il runtime verifica, al caricamento di un assembly, la sua attendibilità sulla base della cosiddetta evidence, calcolata utilizzando parametri come la firma dell'assembly via *strong-name* o certificato digitale, la sua zona di esecuzione, ecc…, utilizzandola per identificare il gruppo di codice di appartenenza e i relativi permessi di esecuzione. I gruppi di codice *(code group)* vengono generalmente determinati dagli amministratori di sistema.
Se i permessi richiesti non coincidono con il gruppo di codice di riferimento, il runtime genera una eccezione di sicurezza.

ApplicationDeployment è la classe più importante del namespace in quanto, oltre a gestire gli aggiornamenti, consente di operare sull'istanza corrente, recuperare le informazioni di deploy, verificare se l'applicazione viene eseguita per la prima volta, fino ad operazioni più complesse come il download di assembly o gruppi di assembly non ancora installati, operazione che vedremo più avanti. La proprietà statica *CurrentDeployment* consente di recuperare l'istanza corrente e di verificare la presenza di nuove versioni utilizzando il metodo *CheckForUpdate()*. Se sono presenti nuove versioni nella cartella di pubblicazione, viene abilitato il pulsante *btnUpdate* che permette di eseguire il codice per l'aggiornamento. Nel gestore dell'evento *click* del pulsante *btnUpdate* inseriamo, quindi, il seguente codice:

```
ApplicationDeployment deploy =
ApplicationDeployment.CurrentDeployment;
deploy.UpdateCompleted += new
                    AsyncCompletedEventHandler(
                        deploy_UpdateCompleted);
deploy.UpdateAsync();
```

Al click sul pulsante recuperiamo l'istanza corrente, impostiamo il gestore per l'evento *UpdateCompleted* e avviamo l'aggiornamento in modalità asincrona richiamando il metodo *UpdateAsync*. È anche possibile effettuare l'aggiorna-

Proprietà	Descrizione
AvailableVersion	Proprietà di tipo *Version* contenente la nuova versione disponibile
IsUpdateRequired	Indica se l'aggiornamento è richiesto per il corretto funzionamento dell'applicazione
MinimumRequiredVersion	Di tipo *Version* indica la versione minima richiesta per poter effettuare l'aggiornamento
UpdateAvailable	Valore booleano che indica se un aggiornamento è disponibile
UpdateSizeBytes	Restituisce la dimensione dei file da aggiornare

Tabella 1: aaa

mento in modalità sincrona utilizzando il metodo *Update* che ritorna *true* se il download e l'installazione è andata a buon fine e *false* se l'operazione non è riuscita. Al termine dell'aggiornamento, possiamo continuare a lavorare con la versione corrente ed ottenere la nuova versione solo alla successiva esecuzione, oppure, sempre attraverso il codice, chiediamo all'utente di riavviare l'applicazione per completare l'aggiornamento. Per ottenere questo risultato inseriamo il seguente codice nel gestore dell'evento *UpdateCompleted:*

```
void deploy_UpdateCompleted(object sender,
                                AsyncCompletedEventArgs e)
  {
      DialogResult result =
          MessageBox.Show("Aggiornamento
              completato! Riavviare l'applicazione?", "",
          MessageBoxButtons.YesNo,
                                MessageBoxIcon.Question);
      if (result == DialogResult.Yes)
      {
```

```
          Application.Restart();
      }
  }
```

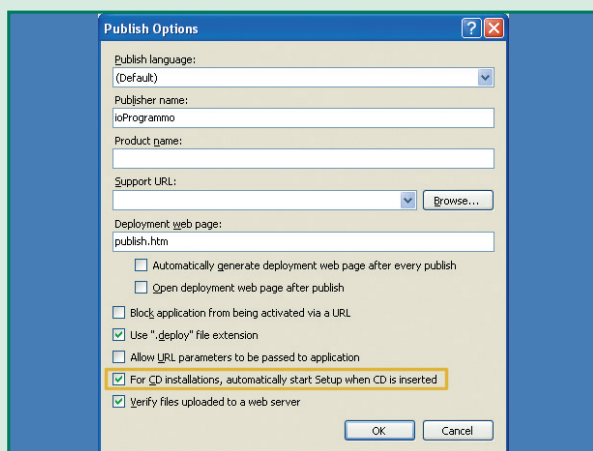
Con veramente poche righe di codice è possibile gestire un'operazione complessa come quella dell'update in tempo reale dell'applicazione sul client. È importante sottolineare, però, che l'applicazione così impostata necessita di richiedere i permessi massimi di esecuzione (*Full Trust*). È anche possibile recuperare le informazioni sulla nuova versione disponibile utilizzando il metodo *CheckForDetailedUpdate()*, che restituisce un'istanza dell'oggetto *UpdateCheckInfo*. L'oggetto contiene le proprietà ruportate in **Tabella 1**.

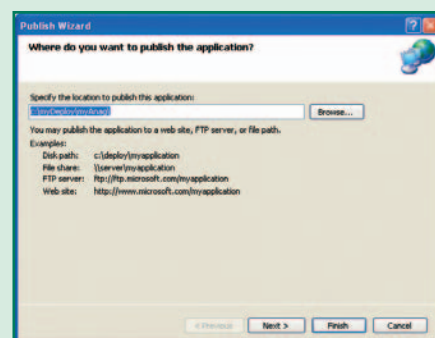
DOWNLOAD ON DEMAND

Nella modalità di distribuzione via internet potremmo trovarci di fronte a client che non dispongono di una larghezza di banda tale da consentire una agevole esecuzione dell'applicazione. In quest'ottica è opportuno, e a volte necessario, minimizzare i file da includere nell'installazione, riducendo quindi il tempo di download e di startup dell'applicazione. *ClickOnce* consente di installare gli assembly o i files anche successivamente, utilizzando le API incluse in *System .Deployment,* ed introducendo il concetto di late-download o download a gruppi. Supponiamo di voler fornire un controllo aggiuntivo, ad esempio un controllo *calendar*, utilizzato dai client solo in alcuni casi e contenuto in un

DISTRIBUIRE IN MODO TRADIZIONALE

La distribuzione delle applicazioni può avvenire anche attraverso supporti come CD o DVD. Questo è possibile

> CREAZIONE DEL FILE AUTORUN.INF

1 Impostiamo nelle Opzioni della scheda Publish delle proprietà del progetto la creazione del file autorun.inf per consentire l'avvio automatico del setup. Possiamo inserire anche altre informazioni quali ad esempio il nome del publisher, il nome del prodotto, un eventuale indirizzo web per il supporto onLine, altre informazioni di vario genere

> SELEZIONE DI UNA CARTELLA DEL FILE SYSTEM

2 Nella prima scheda, impostiamo come destinazione della pubblicazione una cartella sul disco.

assembly esterno alla nostra applicazione. Aggiungiamo un riferimento alla libreria contenente il controllo ed inseriamo il codice necessario per poterlo richiamare (il codice completo è disponibile sul cd allegato alla rivista). Senza pesare sullo startup dell'applicazione è possibile creare gruppi di files e avviare il download dei file non installati solo quando necessario. Facciamo click con il tasto destro sul progetto e clicchiamo su proprietà, richiamiamo la scheda *Publish* e clicchiamo su *ApplicationFiles.* Ci viene mostrata una dialog box contenente l'elenco dei file che saranno distribuiti via *ClickOnce,* ognuno con lo stato di pubblicazione e il gruppo di download di appartenenza. Lo stato di pubblicazione può assumere tre valori: *Include,* indica che il gruppo è incluso dall'installazione di base, *Prerequisite,* che imposta il gruppo come un prerequisito fondamentale per il funzionamento dell'applicazione, e *Exclude,* che esclude il gruppo dall'installazione di base. Esiste solo un gruppo creato di default ed è (*Required*), ma è possibile crearne di nuovi cliccando su *New* ed impostando il nuovo gruppo, che viene poi direttamente assegnato al file selezionato. Utilizzando questa sequenza creiamo il gruppo *controls* e lo assegniamo alla libreria di controlli precedentemente aggiunta alla soluzione. Prima di richiamare ed eseguire il codice contenuto in un controllo appartenente al gruppo *controls,* dobbiamo verificare se gli assembly e i file relativi sono stati già scaricati ed installati sul client. In caso contrario procediamo con il download. Il codice da implementare è simile al seguente:

```
ApplicationDeployment deploy =
ApplicationDeployment.CurrentDeployment;
if (!deploy.IsFileGroupDownloaded("controls"))
{
deploy.DownloadFileGroup("controls");
}
// qui il codice da eseguire
```

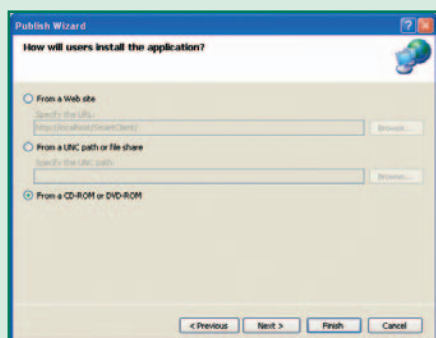
Il codice sopra riportato, verifica se il gruppo denominato *controls* è stato già scaricato dalla cartella di pubblicazione ed eventualmente procede al download della libreria di controlli. Nell'applicazione *myAnag* questo avviene prima della chiamata al form contenente il controllo che andremo ad utilizzare. Al pari della procedura di upload è possibile utilizzare anche il metodo asincrono *DownloadFileGroupAsync* e

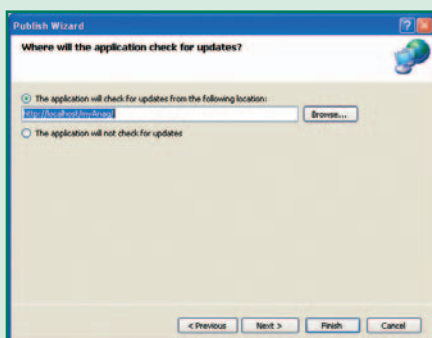
IMPOSTARE MANUALMENTE LE BUILD

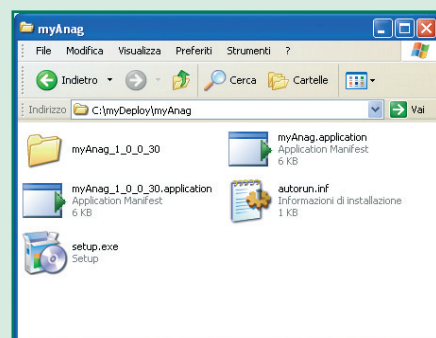
Ad ogni pubblicazione, *ClickOnce* incrementa il numero di build per distinguere le versioni e consentire al runtime di verificare la presenza di aggiornamenti. Spesso però questa procedura genera build non necessarie. È possibile infatti rimuovere questa caratteristica e scegliere di impostare manualmente il numero di build deselezionando il controllo check box per l'incremento automatico della scheda *Publish,* nelle proprietà del progetto da distribuire.

Impostare manualmente la build

sfruttando la distribuzione attraverso file system nel Publish Wizard

> INSTALLAZIONE ATTRAVERSO CD O DVD

3 La seconda scheda chiede la modalità di installazione dell'applicazione. Selezioniamo installazione da cdrom/dvd

> MODALITÀ DI AGGIORNAMENTO

4 Viene chiesto se l'applicazione deve verificare la presenza di aggiornamenti e qual è l'indirizzo di riferimento

> LA CARTELLA DI PUBBLICAZIONE

5 Ecco i file che troviamo nella cartella di pubblicazione e che sarà poi distribuita su CD o DVD.

L'AUTORE

Fabio Cozzolino lavora come responsabile tecnico nello sviluppo di progetti web ed applicazioni distribuite presso la Fimed Srl di Molfetta (BA). È cofondatore di DotNetSide, lo user group del sud italia il cui intento è quello di organizzare eventi di maggior spessore tecnico legati allo sviluppo con il .NET Framework. Possiede la certificazione MCAD.NET. Il suo blog è raggiungibile all'indirizzo http://www.dotnetside.org/blogs/fabio

gestire l'evento *DownloadFileGroupCompleted* per catturare il termine del download.

CLICKONCE E LA CODE ACCESS SECURITY

A differenza delle normali applicazioni .NET, le applicazioni *ClickOnce* seguono un processo di esecuzione leggermente differente. Come potete notare dai link riportati dalla pagina *publish .htm*, viene eseguito un file **.application* che altro non è che il deployment manifest, un file XML dove sono descritte informazioni strettamente legate alla pubblicazione attraverso *ClickOnce* come, ad esempio, l'identità dell'applicazione e come essa deve essere distribuita. Il file *.application* viene eseguito perché associato alla libreria di classi COM *dfshim.dll*, eseguita attraverso la stringa

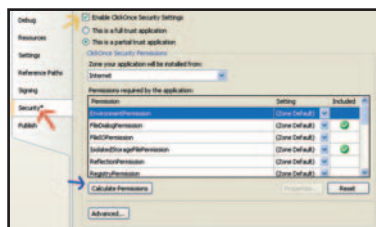
```
rundll32.exe dfshim.dll,ShOpenVerbApplication %1,
```

dove %1 rappresenta, appunto, il percorso completo del file *.application*. Dal manifest viene recuperato il file da eseguire e che sarà sottoposto alle regole di *Code Access Security* (CAS). Per meglio descrivere il meccanismo che muove la CAS sarebbe necessario un articolo dedicato, ma volendo esprimerlo con poche parole, e relativamente all'ambito di nostra competenza, possiamo dire che è il processo di verifica e validazione dei permessi di un assembly dipendentemente dalla zona in cui esso viene eseguito. Come detto anche le applicazioni *ClickOnce* sono sottoposte al controllo della CAS e devono perciò dichiarare quelli che sono i permessi che richiedono per essere eseguiti. Nel nostro caso l'applicazione *myAnag* viene eseguita o installata attraverso la *Internet Zone*, motivo per cui gli vengono garantiti i diritti minimi di esecuzione. Se, viceversa, *myAnag* necessita e richiede diritti di esecuzione superiori a quelli garantiti per la zona, *ClickOnce* chiederà all'utente di confermarne l'esecuzione. Ma se dobbiamo distribuire via internet e la nostra applicazione richiede diritti superiori a quelli possibili con la *Internet Zone*, come possiamo risolvere il problema per evitare la poco gradevole *Security Warning?* Abbiamo due alternative:

1) Distribuire l'applicazione utilizzando supporti removibili come CD o DVD, che ottengono i permessi di una zona *Full-Trust*, pur mantenendo la modalità di aggiornamento via internet;
2) Ottenere un certificato digitale per garantire l'autenticità del codice;

Con *ClickOnce*, infatti, è possibile firmare i file .application utilizzando certificati digitali rilasciati da *Certification Authority* riconosciute. Questo implica, però, un passaggio aggiuntivo che consenta la registrazione del publisher sul computer client, al fine di riconoscerlo come *trusted*. Dopo la registrazione nello store locale, saremo in grado di eseguire correttamente e senza errori la nostra applicazione *ClickOnce*.

CALCOLARE AUTOMATICAMENTE I PERMESSI DI ESECUZIONE

Anche le applicazioni *ClickOnce* sono sottoposte, come abbiamo visto, al controllo della *Code Access Security.* Per questo motivo è importante dichiarare quelli che sono i permessi di esecuzione che si richiedono e che andranno a formare il permission set dell'assembly.
***ClickOnce* consente di impostare sia manualmente, sia utilizzando procedure guidare il livello di sicurezza da applicare. Attraverso la scheda *Security* è possibile, infatti, dichiarare l'applicazione come *Partially Trusted* ed impostare la zona da cui sarà eseguita. *ClickOnce* calcolerà i permessi relativi alle impostazioni date. Se, ad esempio, impostiamo la zona Internet, verranno attribuiti i permessi minimi di esecuzione *(FileDialogPermission, IsolatedStorageFilePermission, SecurityPermission, UIPermission e PrintingPermission).* Ma se l'applicazione richiede in realtà permessi maggiori, come ad esempio, la scrittura sul file system identificata dal FileIOPermission, verrà generata una eccezione.**
In alternativa è possibile calcolare i permessi necessari cliccando sul tasto *CalculatePermission.*
***ClickOnce,* infatti, legge via *reflection* l'albero dei riferimenti e recupera le richieste di permessi del codice referenziato.**

Calcolare i permessi di esecuzione

CONCLUSIONI

Abbiamo visto come con la tecnologia *ClickOnce,* seguendo il caso e l'evolversi di un'applicazione reale, sia possibile eseguire il deploy e l'aggiornamento anche di complesse applicazioni client in modo semplice, facile e sicuro quasi quanto la distribuzione di applicazioni Web. È certo che *ClickOnce* non può essere utilizzato per tutti i tipi di installazione, specialmente quando si rende necessario eseguire operazioni locali complesse come la modifica del file di registro o la registrazione di componenti, modalità consentita solo attraverso l'uso di *Windows Installer.* Il sistema ClickOnce, però, consente senza dubbio di diminuire quel gap, discusso all'inizio di questo articolo, tra la distribuzione di applicativi Windows e la distribuzione di applicativi Web. Per ulteriori chiarimenti vi invito ad utilizzare il forum di ioProgrammo oppure il mio blog riportato sul box laterale. Buona distribuzione.

Fabio Cozzolino

PROGRAMMIAMO IL TASTO "F1"

IN QUESTO ARTICOLO SFRUTTIAMO UN TOOL DI SUN: JAVAHELP PER COSTRUIRE UN SISTEMA DI DOCUMENTAZIONE EFFICACE PER LE NOSTRE APPLICAZIONI. VEDREMO COME GESTIRE L'ALBERO DEGLI ARGOMENTI E COME CREARE COLLEGAMENTI OPPORTUNI

L'idea è molto semplice: realizzare un sistema di documentazione per le proprie applicazioni. Ora, se qualche volta avete premuto il tasto *F1* mentre lavoravate con un software, è probabile che vi sia comparso il famoso *"Aiuto in linea"*. Il punto è che questo *"Help On Line"* per quanto a prima vista possa sembrare privo di funzionalità importanti, ne ha invece alcune che è necessario implementare, se si vuole creare un sistema di documentazione efficace. Come è possibile effettuare una ricerca per argomento? E per parola chiave? Come si indicizza la documentazione? A questo e a molto altro daremo una risposta in questo articolo.

STRUMENTI E PRIMI PASSI

JavaHelp è un sistema di help realizzato in Java proposto da Sun. I concetti che stanno alla base di JavaHelp sono sufficientemente semplici: si scrive la documentazione in HTML utilizzando una particolare logica, si descrive la struttura con cui questi file sono collegati tramite un file XML. In JavaHelp sono poi integrati strumenti per la ricerca di testo ed esistono particolari interfacce per collegare la documentazione a una qualsiasi applicazione Java. Con JavaHelp è possibile costruire a mano un intero sistema di Help, difatti sia i documenti HTML che quelli XML sono facilmente editabili con qualsiasi programma per elaborazione di testo, compreso l'onnipresente notepad o un suo qualsiasi clone. Se tuttavia non si desidera avere a che fare direttamente con XML si possono usare anche appositi software. L'uso di strumenti particolari è consigliabile quando la dimensione del help cresce e diventa sempre più difficile gestire manualmente i riferimenti e i collegamenti tra i documenti.

LA LISTA DELLA SPESA

Scarichiamo prima di tutto il package Java Help. Aprite la pagina all'indirizzo *http://java.sun.com/products/javahelp/* e seguite i link che portano alla versione JavaHelp 2.0_02, l'ultima rilasciata al momento della stesura dell'articolo. Scaricate il file JavaHelp *2.0_02.zip,* comprensivo della libreria e della documentazione e scompattatelo in una cartella di vostro gradimento. Il primo passo per redigere l'help è quello di provvedere al contenuto. Creiamo quindi una cartella che chiameremo *"help"* che conterrà tutti un insieme di file che costituiranno quello che chiameremo un *"help set"*. All'interno di questa cartella, a titolo puramente esemplificativo, creiamo le due directory chiamate *"argument1"* e *"argument2"*. In ognuna di queste cartelle creiamo 3 file HTML con nomi rispettivamente da *"item000 .html"* a *"item003.html"* nella prima e da *"item004 .html"* a *"item006.html"* nella seconda. I documenti HTML del help set devono avere la solita struttura con header e body.

```
<html>
  <head>
    <title>[...]</title>
  </head>
  <body>
    [...]
  </body>
</html>
```

Per l'esempio è sufficiente riempire il body con del testo qualsiasi. Lasciamo al lettore il compito di riempire il contenuto del file di help. Nei sorgenti che vengono forniti con l'articolo comunque i file di help di esempio sono compilati. L'unica nota da tenere in mente nella stesura del contenuto degli help è di utilizzare percorsi relativi nel caso si inserisca un link tra un documento e l'altro. Se ad esempio si desiderasse inserire un link dal documento *"item 000.html"* nella cartella *"argument1"* al file *"item004 .html"* nella cartella *"argument2"* sarebbe necessario specificare un link siffatto.

```
<a href="../argument2/item004.html">text</a>
```

Questa precauzione permette di spostare l'help set in

qualsiasi directory senza pregiudicare la navigazione tra i documenti, a patto ovviamente di mantenere la struttura delle directory dell'help set costante.

NOMINARE I DOCUMENTI

La mappa del help set è il documento XML più importante del nostro progetto. Il file consente di assegnare una sorta di etichetta identificativa ad un file determinando una corrispondenza univoca tra file HTML e nomi. Nell'intero file di Help sarà dunque possibile riferirsi a un file utilizzando l'etichetta che gli è stata associata piuttosto che il suo percorso. Questo permette da un lato di risparmiare tempo per la digitazione dei nomi dei file, oppure di effettuare delle variazioni semplicemente modificando il puntatore associato a un file nella descrizione XML. L'unico vincolo riguarda l'estensione del file della mappa, è necessario infatti che tale file abbia estensione *".jhm"*. Per procedere create quindi il file *"my_map.jhm"* come riportato di seguito.

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE map PUBLIC "-//Sun Microsystems
        Inc.//DTD JavaHelp Map Version 1.0//EN"
 "http://java.sun.com/products/javahelp /map_1_0.dtd">
<map version="1.0">
  <mapID target="item001" url=
              "argument1/item001.html" />
  <mapID target="item002" url=
              "argument1/item002.html" />
  <mapID target="item003" url=
              "argument1/item003.html" />
  <mapID target="item004" url=
              "argument2/item004.html" />
  <mapID target="item005" url=
              "argument2/item005.html" />
  <mapID target="item006" url=
              "argument2/item006.html" />
</map>
```

Il documento è di palese interpretazione. Ad esempio al file *"section001/item001.html"* è stato assegnato il nome *"item001"*.

DEFINIRE LA TOC

TOC è l'acronimo di *Table of Content*. È una struttura che elenca tutti gli argomenti inseriti nell'help e permette un accesso rapido ad essi mediante un menu ad albero. Per definire la TOC create il file *"my_toc.xml"*. Anche in questo caso il nome del file è opzionale. Digitate il seguente contenuto:

```
<?xml version='1.0' encoding='UTF8' ?>
<!DOCTYPE toc PUBLIC "-//Sun Microsystems Inc.
        //DTD JavaHelp TOC Version 2.0 //EN" "http:
     //java.sun.com/products/javahelp /toc_2_0.dtd">
<toc version="2.0">
  <tocitem text="help">
    <tocitem text="Section 001">
      <tocitem text="Item 001" target="item001" />
      <tocitem text="Item 002" target="item002" />
      <tocitem text="Item 003" target="item003" />
    </tocitem>
    <tocitem text="Section 002">
      <tocitem text="Item 004"  target="item004" />
      <tocitem text="Item 005"  target="item005" />
      <tocitem text="Item 006"  target="item006" />
    </tocitem>
  </tocitem>
</toc>
```

Il file è di semplice comprensione. L'elemento esterno *"toc"* definisce l'inizio e la fine del tavola dei contenuti. Ogni elemento della toc è definito da un elemento *"tocitem"*. Un *tocitem* può contenere al suo interno un qualsiasi numero di *tocitem*. Questo permette di rappresentare una qualsiasi struttura ad albero, ramificata a piacere. Quando l'elemento *tocitem* presenta l'attributo *target*, la finestra di help farà sì che cliccando sull'elemento grafico rappresentante quell'elemento, venga visualizzata la pagina il cui nome è quello specificato nell'attributo *target* dell'elemento *tocitem*. Ad esempio, con la toc che abbiamo usato come esempio, il nodo principale dell'albero sarà definito dall'etichetta *"help"*. Cliccandolo si apriranno due sottonodi, rispettivamente indicati dalle etichette *"Section 001"* e *"Section 002"*. Aprendo il primo, saranno visualizzati altri tre nodi, *"Item 001"*, *"Item 002"* ed *"Item 003"*. Cliccando ad esempio *"Item 002"*, il sistema di help preleva l'attributo target del *tocitem* pari a *"item002"*. Attraverso il file

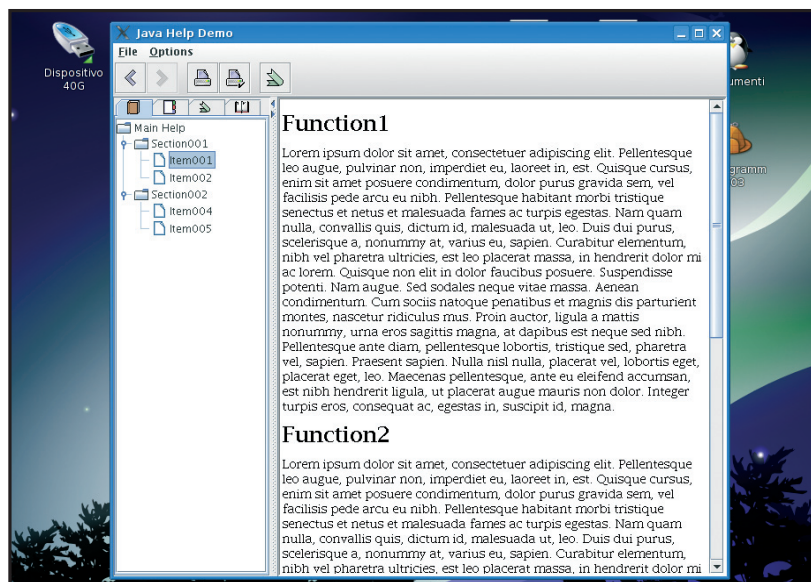



Fig.1: Un esempio di come abbiamo strutturato il file di help

map il sistema preleva il file corrispondente al nome *"item002"* che è *argument1/item002.html* e lo visualizza.

COMBINARE IL TUTTO IN UN HELP

Questa struttura non ci servirà a niente se non riuniamo tutto insieme per farne un Help completo. Abbiamo bisogno ancora di un file XML con estensione *".hs"* che definirà l'help set. In questo file sono inclusi i riferimenti al file Map e a tutti i file accessori da utilizzare. L'esempio seguente mostra il file *my_helpset.hs,* che definisce questo primo esempio di help.

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE helpset
PUBLIC "-//Sun Microsystems Inc.//DTD JavaHelp
                                  HelpSet Version 1.0//EN"
        "http://java.sun.com/products/javahelp
                                        /helpset_1_0.dtd">
<helpset version="1.0">
```

Il primo elemento *"title"* assegna un nome all'help che verrà visualizzato nella finestra dell'help.

```
  <title>Java Help Demo</title>
```

L'elemento maps serve a specificare quali sono le mappe che definiscono i nomi dei documenti utilizzati dal sistema di help. Inseriamo in questo elemento il nome del file da noi creato.

```
  <maps>
     <homeID>intro</homeID>
     <mapref location="my_map.jhm"/>
  </maps>
```

Il file definisce una vista per ogni struttura d'accesso. Ad esempio il seguente caso mostra come configurare la vista per la TOC. Sostanzialmente si configura l'help in modo che la TOC venga visualizzata attraverso la classe *javax.help.UiteAppendMerge* fornita con Java Help

```
  <view mergetype="javax.help.UniteAppendMerge">
    <name>TOC</name>
    <label>Table Of Contents</label>
    <type>javax.help.TOCView</type>
    <data>my_toc.xml</data>
  </view>
```

Il file termina ovviamente con le chiusure dei tag ancora aperti.

```
</helpset>
```

VISUALIZZARE L'HELP

Per visualizzare l'help appena costruito utilizziamo una classe di utility fornita con Java Help. Portatevi nella cartella dove avete decompresso lo zip scaricato dal sito Sun e lanciate il seguente *jar.*

```
jh2.0\demos\bin\hsviewer.jar
```

Si aprirà una finestra che vi chiederà di aprire un file *.hs* di definizione dell'helpset. Cliccate su *"browse"* e navigate nel filesystem fino a selezionare il file *my_helpset.hs* appena creato. Vi si aprirà una finestra in cui potrete navigare e verificare il funzionamento dell'help appena creato.

INCLUDERE L'HELP IN UN'APPLICAZIONE

Ovviamente un help indipendente da un'applicazione ha un'utilità limitata. Vediamo ora come sia possibile aggiungere alle nostre applicazioni Java stand alone l'accesso ad un help realizzato con JavaHelp. L'applicazione consisterà in un'applicazione swing stand alone con una menu bar con una voce che avrà come unico compito quello di visualizzare l'help appena costruito. La nostra applicazione dimostrativa è costituita da una semplice classe costituita da un unico Jframe. Estendiamo quindi questa classe. In più la classe *Help* che stiamo scrivendo implementa anche l'interfaccia *ActionListener* in modo da poter gestire l'unica voce di menu prevista e attivare in quel momento l'help.

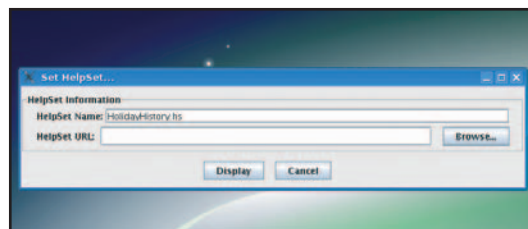



Fig.2: **hsviewer ci chiede di selezionare il file di help da visualizzare**

La prima sezione della classe importa il package *javax.help,* le cui classi sono contenute nel file *jar* che accompagna la distribuzione di JavaHelp.

```
package it.ioprogrammo;
import java.awt.event.*;
import java.net.URL;
import javax.help.*;
import javax.swing.*;
public class Help extends JFrame implements
                                        ActionListener {
```

Scriviamo un metodo *buildMenuBar()* che si occuperà della creazione della barra di menu. Il procedi-

mento è abbastanza lineare. Viene creato un oggetto *Build* di classe *JMenuBar* e che rappresenta la nostra barra. Viene creato un oggetto *helpMenu* di classe *JMenu* che rappresenta un menu, infine il menu viene associato alla barra tramite il metodo *Add.* Non ci resta che aggiungere delle voci al menu tramite un *HelpItem* di classe *JMenuItem.* Si noti che a quest'ultimo, il cui riferimento è contenuto nella variabile *helpItem,* viene associato come action listener l'istanza stessa. Questo ci permetterà di gestire semplicemente la visualizzazione dell'help senza ricorrere a inner class varie.

```
private JMenuBar buildMenuBar() {
    JMenuBar built = new JMenuBar();
    JMenu helpMenu = new JMenu();
    helpMenu.setText("Help");
    built.add(helpMenu);
    helpItem = new JMenuItem();
    helpItem.setText("Show");
    helpItem.addActionListener((ActionListener) this);
    helpMenu.add(helpItem);
    return built;
}
```

Il metodo di costruzione del menu viene richiamato all'interno del costruttore. In questo modo ogni applicazione istanziata avrà automaticamente costruito il menu. Vengono anche impostati il tiolo della finestra e le relative dimensioni tramite i metodi *setTitle()* e *setSize()*.

```
Help() {
    super();
    this.setTitle("Using JavaHelp");
    this.setSize(300, 300);
    this.setJMenuBar(buildMenuBar());
}
```

VISUALIZZARE L'HELP

Veniamo al metodo centrale della classe, quello per la visualizzazione dell'help. Il metodo in cui attiviamo la visualizzazione è *actionPerformed()* poiché questo viene richiamato ogni volta che l'utente clicca sulla voce di menu. Viene dichiarato un oggetto di tipo *JHelp.* Questa classe, fornita con JavaHelp, permette di visualizzare un help a partire dalla definizione contenuta nel file *.hs.* Come prima cosa si ottiene il *ClassLoader* della nostra applicazione di esempio. Poi serve l'URL dell'HelpSet creato nei passi precedenti. Per trovarlo si utilizza il metodo statico di utilità *findHelpSet* della classe *HelpSet.*
Questo metodo controlla se il *ClassLoader* passatogli è in grado di trovare la risorsa con il percorso specificato, e in caso affermativo lo restituisce. Se tutto ha funzionato possiamo istanziare un nuovo HelpSet con i dati appena trovati. Ciò si ottiene con il costruttore *HelpSet()* a cui vengono passati il classloader e l'URL. JHelp è sostanzialmente un componente che mostra un HelpSet e può essere visualizzato all'interno di un Pane. Istanziamo quindi il JHelp passandogli il riferimento all'HelpSet da visualizzare, ottenendo così finalmente un riferimento al visualizzatore del help. I passi seguenti sono abbastanza comuni in ogni applicazione che faccia uso di swing. Costruiamo una nuova finestra e impostiamone il *content pane* con il riferimento al visualizzatore di help appena ottenuto e impostiamone la proprietà *visibel* a *true* in modo da farla apparire.

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource().equals(helpItem)) {
        JHelp helpViewer = null;
        try {
            ClassLoader cl =
                        this.getClass().getClassLoader();
            URL url = HelpSet.findHelpSet(
                  cl, "it/ioprogrammo/my_helpset.hs");
            helpViewer = new JHelp(new HelpSet(cl, url));
        } catch (Exception e1) {
            e1.printStackTrace();}
        JFrame frame = new JFrame();
        frame.setSize(500, 500);
        frame.getContentPane().add(helpViewer);
        frame.setDefaultCloseOperation(
                        JFrame.DISPOSE_ON_CLOSE);
        frame.setVisible(true);}
}
```

Realizziamo ora un semplice metodo *main()* in modo da rendere la classe utilzzabile come programma stand alone. Di seguito il codice del metodo *main.* Come possibile vedere il metodo si limita ad istanziare una finestra Help e a renderla visibile tramite il metodo *setVisible()*.

```
public static void main(String[] args) {
    Help window = new Help();
    window.setVisible(true);
}
```

CONCLUSIONI

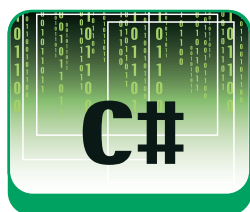
L'articolo ha mostrato come utilizzare JavaHelp per realizzare un semplice sistema di help e come innestarlo all'interno delle nostre applicazioni. Le potenzialità di JavaHelp sono tuttavia maggiori. Ad esempio oltre alla TOC è possibile aggiungere al help una struttura ad indice, un glossario ed anche una ricerca testuale con indicazione di un ranking per la pagina proposta. Insomma è possibile creare un help professionale.

Daniele De Michelis

COMPILARE E LANCIARE L'APPLICAZIONE

Si compili l'applicazione avendo cura di aggiungere al CLASSPATH le librerie proprie di *JavaHelp jh.jar, jhall.jar, jhbasic.jar, jsearch.jar* presenti nella sottocartella */javahelp/lib* della directory nella quale avete scelto di scompattare il file *.zip.*
Anche per lanciare l'applicazione è necessario avere cura di includere le suddette librerie nel *CLASSPATH.*

WINDOWS WORKFLOW FOUNDATION

OGNI APPLICAZIONE, PICCOLA O GRANDE CHE SIA, IMPLEMENTA UN WORKFLOW. IN QUESTO ARTICOLO SCOPRIREMO COME CREARE WORKFLOW, ANCHE COMPLESSI, CON LA NUOVA TECNOLOGIA MICROSOFT NATA PER GESTIRE FACILMENTE I FLUSSI

Prima di addentrarci nella tecnologia *Windows Workflow Foundation,* è bene capire cosa è un *workflow* partendo dalla sua definizione. Un *workflow* è la formale, ma sintetica, descrizione di un processo costituito da una serie di task (attività elementari), eventualmente cicliche o alternative, da eseguire per ottenere un preciso risultato. Il concetto è abbastanza semplice se calato in un contesto reale e magari quotidiano: per navigare su internet dobbiamo:

1. accendere il PC;
2. attendere che il sistema operativo si carichi;
3. eseguire il logon;
4. se il logon ha esito positivo dobbiamo avviare la connessione;
5. aprire il browser;
6. digitare l'url.

I sei passaggi appena elencati sono, a tutti gli effetti, "una sequenza di task elementari da eseguire per ottenere un preciso risultato". Un *Workflow* insomma! È facile immaginare che, in ambito software, si sia continuamente circondati da *workflow* sia semplici sia decisamente complessi. Si provi ad esempio ad immaginare un sito web di e-commerce. Sono decine, se non centinaia, i *workflow* che governano le attività di un sito web di tale tipo. *Workflow* che, consapevolmente o meno, abbiamo già implementato. Proviamo ad immaginare il *workflow* che conduce all'ordine di un prodotto: uno dei task potrebbe essere la selezione della merce da acquistare (un task ciclico), un altro la scelta dell'indirizzo di spedizione, un altro ancora il pagamento dell'importo dovuto etc. Task che, come ad esempio quello relativo al pagamento, possono al loro interno contenere interi *workflow!* Compreso il concetto di *Workflow,* vediamo cosa ci propone Microsoft con *Windows Workflow Foundation.*

Fig. 1: ***I template di Visual Studio 2005 per Workflow***

WINDOWS WORKFLOW FOUNDATION

L'idea è molto semplice, ovvero quella di fornire all'utente una serie di strumenti per disegnare graficamente il "workflow" di un'applicazione. La rivoluzione sta nel fatto che automaticamente in perfetta corrispondenza al disegno verrà generato il codice che consente al programmatore di implementare un task. *Windows Workflow Foundation* è un framework per lo sviluppo di workflow in ambiente Windows. È costituito sia da API sia di tool integrati in Visual Studio 2005 per lo sviluppo e l'esecuzione di applicazioni basate su *Workflow.* Sin dalla versione 2003 di Visual Studio infatti, Microsoft ha portato avanti la formula dell'ambiente unico di sviluppo. *Workflow Foundation* non fa eccezione, a tutto vantaggio della produttività. Esattamente come accade nello sviluppo tradizionale quando si crea un'applicazione in modo visuale e la logica applicativa risiede nel code behind, così un workflow viene letteralmente "disegnato" in Visual Studio 2005, trascinando i task direttamente nel designer, mentre la logica risiede in classi separate. Un workflow così creato ha la possibilità di essere usato sia su applicazioni client sia su applicazioni server, Windows forms o asp.net ed in tutte i tipi di applicazioni .net *(managed).* In Visual Studio 2005, abbiamo a disposizione una serie di template per la realizzazione delle diverse tipologie di workflow (**Figura 1**):

- **Sequential Workflow Console Application:** crea un progetto che contiene un workflow se-

quenziale ed un'applicazione console di test.

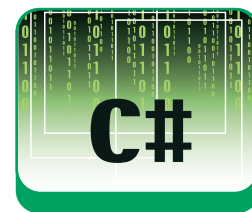
- **Sequential Workflow Library:** crea un progetto che contiene un Workflow sequenziale.
- **Workflow Activity Library:** crea un progetto per la creazione di attività *(task)* da poter utilizzare in altri progetti workflow.
- **State Machine Console Application:** crea un progetto per la realizzazione di una macchina a stati e la relativa console di test.
- **State Machine Workflow Library:** crea un progetto *State Machine*.
- **Empty Workflow:** crea un progetto workflow vuoto.

Come si evince dall'elenco, Visual Studio 2005 supporta 2 diverse tipologie di Workflow: quello sequenziale e la macchina a stati. Un *workflow sequenziale* è caratterizzato da una serie di task in sequenza appunto, ovvero quando il task precedente è terminato inizia il successivo. Sebbene sia di tipo sequenziale, questo workflow può ricevere degli input esterni che, in sostanza, potrebbero modificare l'originale ordine di esecuzione. Una *macchina a stati* invece è costituita da una serie di "stati", uno dei quali sarà lo stato iniziale ed un altro quello finale che denota il completamento del workflow. Dopo aver compreso questi concetti, passiamo subito alla realizzazione del nostro primo Workflow.

COSA CI SERVE PER CREARE WORKFLOW

Sviluppare workflow oggi, con *Windows Workflow Foundation*, non è immediato. La piattaforma su cui tutto il sistema si basa è ancora in beta e gli strumenti di sviluppo non sono ancora completi. Ci tocca quindi scaricare ed installare manualmente diversi componenti. La prima operazione da fare, se abbiamo versioni beta di Visual Studio 2005 installate sul nostro PC, è toglierle. I tool per lo sviluppo di Workflow funzionato solo sulla versione finale di Visual Studio 2005 o sulle versioni Express (sempre finali). Per rimuovere correttamente tutti i componenti in beta, ci viene in aiuto un comodo tool (vedi box laterale). Una volta avviato, sarà lui a selezionare tutti i componenti in beta da rimuovere dalla nostra macchina di sviluppo in modo che essa sia pronta all'installazione dei componenti definitivi ed alle estensioni di *WinFX*. Una volta installata la versione finale di uno dei tool di sviluppo, è necessario scaricare ed installare, nell'ordine:

1. il runtime di *WinFX;*
2. il *Windows SDK;*
3. i *Development Tools* per *WinFX;*
4. le estensioni di *Workflow Foundation* per Visual Studio 2005.

I link per il download di tutto il materiale sono riportati nei box laterali. Una volta installato il tutto, siamo pronti per creare i nostri Workflow.

IL NOSTRO PRIMO WORKFLOW

Per iniziare a prendere un po' di dimestichezza con *Windows Workflow Foundation*, creiamo il nostro primo, semplice, workflow come schematizzato in **Figura 2**.

Inizieremo creandolo per la console di test.

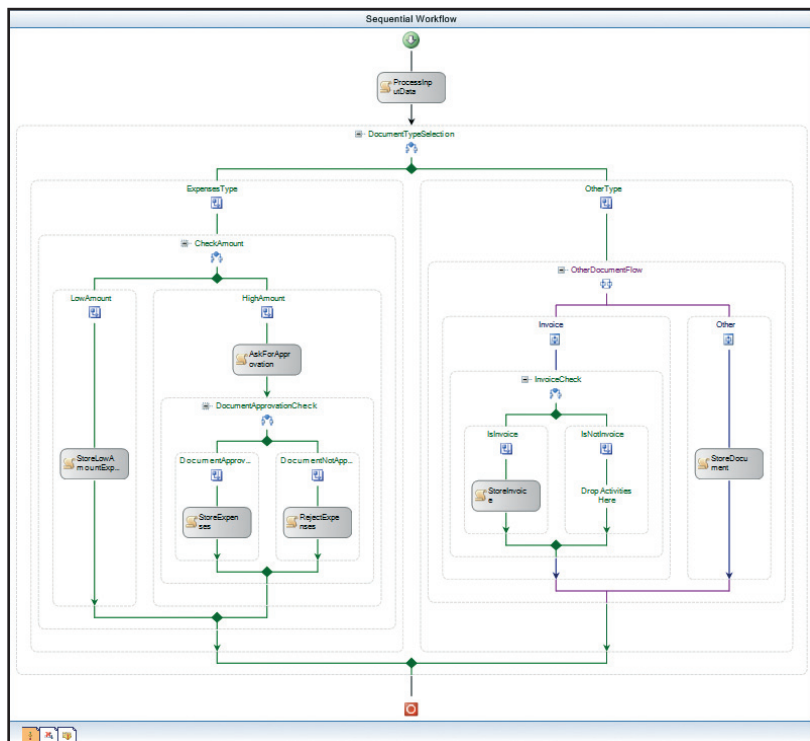
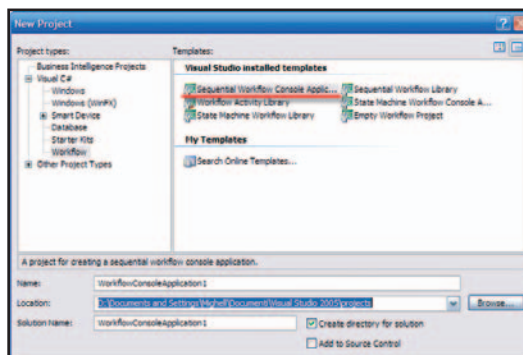
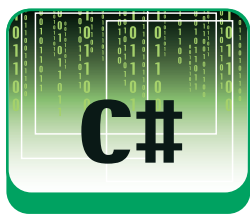
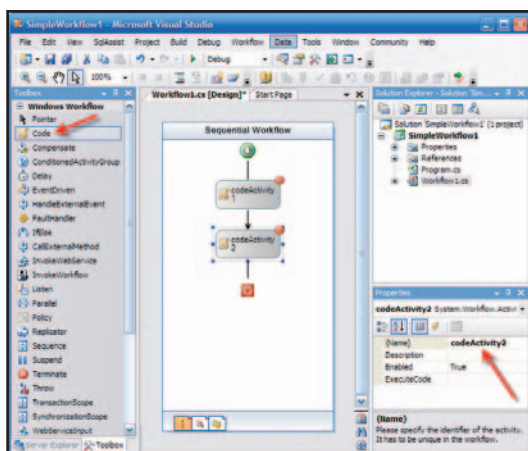



Fig. 2: *Il Workflow completo che realizzeremo*

1 La prima operazione da svolgere è quella di selezionare il tipo di Workflow che vogliamo realizzare. Apriamo quindi Visual Studio 2005 e selezioniamo *File/New Project*. Dalla finestra visibile in figura, selezioniamo la voce *"Sequential Workflow Console Application"* , scegliamo un nome per l'applicazione e specifichiamo un percorso in cui salvarla.

DOWNLOAD

Per scaricare tutto l'occorrente per sviluppare Workflow, accedere alla pagina

http://msdn.microsoft.com/windowsvista/getthebeta/default.aspx

e scaricare l'occorrente.

2 Completato il primo passo, ci troveremo d'avanti un workflow vuoto in cui possiamo inserire i nostri elementi *(Activity)* trascinandoli dalla Toolbox di sinistra. Come è possibile vedere in figura, abbiamo diversi tipi di *Activity* tra cui possiamo scegliere. Per questo primo esempio, scegliamo quella denominata *"Code Activity"* e trasciniamola sul workflow vuoto. Ripetiamo lo stesso passaggio una seconda volta per creare 2 *activity* diverse. Il risultato deve essere simile a quello in figura.

NOTA

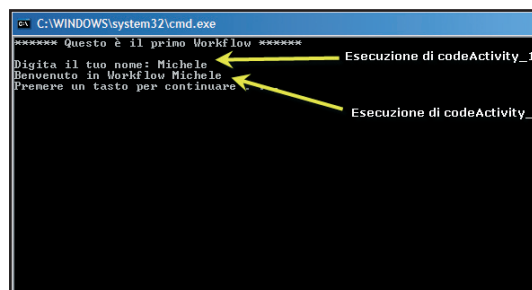
Uno dei siti di riferimento relativo a Windows Workflow Foundation è

http://www.windowsworkflow.net/

ricco di esempi e activity pronte da scaricare.

3 Il nostro *Workflow* contiene ora 2 *task* da eseguire denominati *codeActivity_1* e *codeActivity_2*. Sta a noi ora implementare il codice che definisce il comportamento di questi 2 *task*. Per farlo, è sufficiente fare un doppio clic su ogni *activity* per definirne il comportamento in corrispondenza dell'evento *codeActivityX_ExecuteCode*. Da notare la somiglianza che c'è, a livello di utilizzo dell'editor, con la definizione del comportamento di un pulsante! Definiamo quindi prima una variabile privata di tipo stringa che chiameremo *Name* che valorizzeremo con l'input dell'utente. Tale variabile sarà poi utilizzata da *codeActivity_2* quando verrà eseguita, ovvero al completamento della *codeActivity_1*.

```
public sealed partial class Workflow1:
                                SequentialWorkflowActivity
{   private string Name;
    /// <summary>
    /// Costruttore della classe Workflow1
    /// </summary>
    public Workflow1(){
          InitializeComponent(); }
    /// <summary>
    /// Esecuzione della prima Activity
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void codeActivity1_ExecuteCode(
                   object sender, EventArgs e ) {
        Console.WriteLine( "****** Questo è il primo
                       Workflow ******\r\n" );
        Console.Write( "Digita il tuo nome: " );
        Name = Console.ReadLine();}
    /// <summary>
    /// Esecuzione della seconda Activity
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void codeActivity2_ExecuteCode( object
                            sender, EventArgs e ) {
        Console.WriteLine( "Benvenuto in Workflow
                                {0}", Name ); }
}
```

4 Compiliamo e mandiamo in esecuzione il nostro progetto. Trattandosi di un *"Sequential Workflow Console Application"*, vedremo apparire a video una console che ci chiederà di inserire il nostro nome. È stato eseguito il primo blocco *(codeActivity_1)*. Una volta fatto, premendo il tasto invio, ci troveremo il saluto personalizzato, dato dall'esecuzione del secondo blocco *(codeActivity_2)*.
Il nostro primo workflow è stato eseguito correttamente.

L'esempio, per quanto semplice ci dà la misura di quanto sia immediato implementare un semplice workflow sequenziale. L'aver scelto di usare un progetto di tipo *Sequential Workflow Console Application* ci aiuta notevolmente implementando già le istruzioni per avviare il workflow ed attenderne il risultato.
Se, in Visual Studio 2005, apriamo il file *Program.cs* vedremo il seguente codice:

```
namespace SimpleWorkflow1
{ class Program
  { static void Main(string[] args)
    { WorkflowRuntime workflowRuntime = new
                          WorkflowRuntime();
        AutoResetEvent waitHandle = new
                      AutoResetEvent(false);
        workflowRuntime.WorkflowCompleted +=
                     delegate(object sender,
              WorkflowCompletedEventArgs e)
                      {waitHandle.Set();};
        workflowRuntime.WorkflowTerminated +=
                     delegate(object sender,
             WorkflowTerminatedEventArgs e)
```

```
        { Console.WriteLine(e.Exception.Message);
          waitHandle.Set(); };
        WorkflowInstance instance =
                  workflowRuntime.CreateWorkflow(
               typeof(SimpleWorkflow1.Workflow1));
        instance.Start();
        waitHandle.WaitOne(); } }
}
```

Come prima istruzione troviamo quella relativa all'istanza del *WorkflowRuntime* che rappresenta appunto in runtime in cui "vive" il nostro *Workflow*.
Segue quindi la creazione di *waitHandle* di tipo *AutoResetEvent* il cui scopo è quello di mantenere il thread in attesa finché non interviene un evento che porta il thread nello stato di *"signaled"* tramite la chiamata *waitHandle.Set()*. Subito dopo vengono creati due delegate per la notifica del completamento e della "terminazione" del *Workflow*. Come è evidente dal codice, la differenza tra lo stato di *"WorkflowCompleted"* e quello di *"WorkflowTerminated"* è che nel primo caso il *workflow* è terminato correttamente (lo stato del thread passa in *signaled* attraverso *waitHandle.Set()*, mentre nel secondo il *Workflow* è terminato per un errore. Abbiamo tutti gli elementi per usare il *Workflow* appena creato quindi, ne recuperiamo una istanza com *workflowRuntime.CreateWorkflow(……)* è la facciamo partire con *instance.Start()*. Per finire, *waitHandle.WaitOne()* mette il thread in attesa di ricevere un segnale. Il nostro *Workflow* è partito ed è pronto per eseguire la prima *activity*.

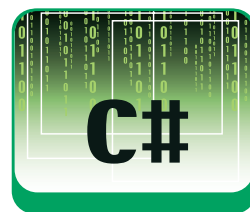
UN WORKFLOW PIÙ COMPLESSO

Nell'esempio fatto in precedenza, abbiamo creato un semplice workflow sequenziale con 2 "activity" il cui scopo era semplicemente quello di chiederci il nome e rispondere con un saluto personalizzato. Il tipo di Workflow scelto conteneva già una console di test. Vedremo ora un workflow leggermente più complesso utile a comprendere meglio le potenzialità di questo strumento. Per testarlo useremo un client Windows rendendo l'esempio più simile alla realtà. A titolo di esempio realizzeremo un Workflow utile all'archiviazione di un documento in funzione della sua tipologia (fattura, rimborso spese etc.). In un caso particolare, andremo anche a valutare una condizione specifica e chiedere una approvazione. Essendo un esempio ed esulando dallo scopo di questo articolo, non è implementata la logica di archiviazione del documento. Una volta compreso il meccanismo del workflow, sarà semplice inserirla in un secondo momento. Come abbiamo fatto per il Workflow precedente, apriamo il nostro Visual Studio 2005 e creiamo due nuovi progetti: il primo sarà il nostro workflow mentre il secondo rappresenterà il client di test.
Come prima cosa, nel progetto Workflow, creiamo una serie di membri che rappresenteranno il nostro documento. Tali membri saranno privati ed esposti attraverso delle proprietà pubbliche:

```
private string _DocumentType;
private string _DocumentTitle;
private string _DocumentOwner;
private int _ExpensesAmount;
public bool Approved;
public string DocumentType {
    get { return _DocumentType; }
    set { _DocumentType = value; } }
public string DocumentTitle {
    get { return _DocumentTitle; }
    set { _DocumentTitle = value; }}
public string DocumentOwner {
    get { return _DocumentOwner; }
    set { _DocumentOwner = value; } }
public int ExpensesAmount {
    get { return _ExpensesAmount; }
    set { _ExpensesAmount = value; }
}
```

Nel progetto Workflow, trasciniamo la prima activity chiamandola *ProcessInputData* il cui scopo reale sarà quello di effettuare una valutazione dei dati ricevuti prima di procedere con il workflow. Nel nostro caso specifico, restituirà solo un messaggio all'utente a conferma della ricezione dei dati:

```
private void ProcessInputData_ExecuteCode( object
                          sender, EventArgs e ) {
    MessageBox.Show( "Dati ricevuti. Inizio il
                                    workflow" );
}
```

Aggiungiamo ora un nuovo tipo di *activity* al nostro workflow di tipo *IfElse*. È facile immaginare che lo scopo di questa particolare activity è quello di valutare una condizione If e scegliere di conseguenza il ramo del *workflow* corretto. Nel nostro caso, la nostra scelta sarà sul tipo di documento ricevuto. Se si tratta di una richiesta di rimborso spese *(Expenses)*, verrà seguito un ramo che, in funzione dell'ammontare della richiesta, richiede una approvazione o meno. Diversamente verrà seguito il secondo ramo.
Per settare la condizione dell'activity *IfElse* selezioniamo uno dei due rami e, dalla finestra delle proprietà, alla voce *Condition*, selezioniamo la voce *System.Workflow.Activities.Rules.RuleConditionReference*. Diamo innanzitutto un nome alla nostra "condizione" e, selezionando la voce *Expression*, apriamo l'editor delle condizioni che ci permette di scrivere la condizione da valutare

Il tool di rimozione delle versioni beta di Visual Studio 2005 è scaricabile da qui:

http://lab.msdn.microsoft.com/vs2005/uninstall/preRTMuninstall/default.aspx

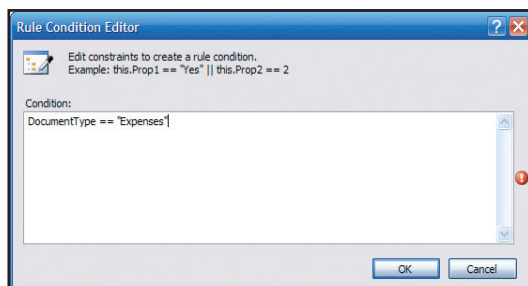
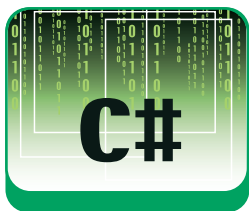



Fig. 3: **L'editor per la selezione delle condizioni della activity IfElse**

nel nostro *If.* Procediamo aggiungendo le varie activity ed impostando le relative condizioni come mostrato in **Figura 3** completando così il nostro workflow.
Non ci resta che realizzare il nostro client per "consumare" il workflow.
Apriamo il form1 del relativo progetto ed inseriamo gli elementi come in **Figura 4**:

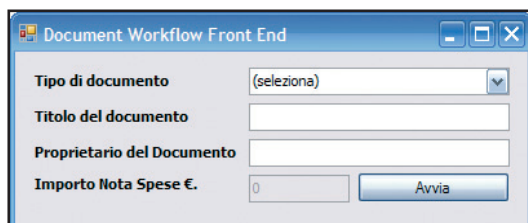



Fig. 4: **Il form per l'utilizzo del nostro workflow**

Non ci resta ora che inserire gli elementi utili ad avviare ed usare il workflow. Come prima cosa, abbiamo la necessità di aggiungere tutti i riferimenti utili al runtime di *Windows Workflow Foundation*. Cliccando con il tasto destro del mouse sulla cartella *"References"* del nostro progetto, aggiungiamo i seguenti componenti: *System.Workflow.Activities*, *System.Workflow.ComponentModel*, *System.Workflow.Runtime* e naturalmente il riferimento al nostro workflow nell'esempio chiamato *DocumentProcess*.
La prima operazione da fare nel nostro form è quella istanziare ed avviare il runtime di *Workflow Foundation:*

```
WorkflowRuntime _wfRuntime;
WorkflowInstance _wfInstance;

public Form1()
{
    InitializeComponent();
    _wfRuntime = new WorkflowRuntime();
    _wfRuntime.StartRuntime();
}
```

Ora che abbiamo tutti i riferimenti, non ci resta che avviare il workflow passando i relativi parametri ed attenderne l'esecuzione. Per farlo, inseriamo il seguente codice nell'evento *click* del bottone:

```
private void btnStartWorkFlow_Click( object sender,
                                         EventArgs e ) {
    Dictionary<string, object> parameters = new
                              Dictionary<string, object>();
    parameters.Add( "DocumentType",
           cmbDocumentType.SelectedItem.ToString() );
    parameters.Add( "DocumentTitle",
                          tbxDocumentTitle.Text );
    parameters.Add( "DocumentOwner",
                          tbxDocumentOwner.Text );
    parameters.Add( "ExpensesAmount",
                int.Parse(tbxExpensesAmount.Text));

    _wfInstance = _wfRuntime.CreateWorkflow(
              typeof( Workflow1 ), parameters );
    _wfInstance.Start();
}
```

Il Workflow partirà usando i parametri passati e seguirà il flusso che abbiamo creato. Il risultato sarà il seguente:

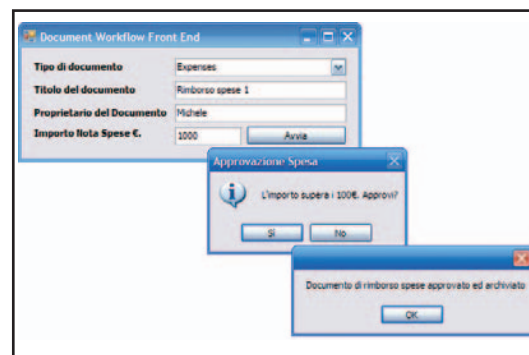
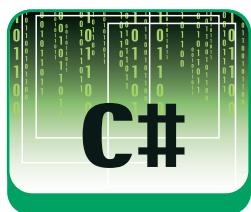



Fig. 5: **Il workflow eseguito**

CONCLUSIONI

In questo numero abbiamo trattato solo una piccola parte di *Windows Workflow Foundation*, ma già si capisce quanto sia potente e comoda questa tecnologia. La possibilità di creare delle *custom activity*, di hostare il *Workflow* come servizio web, di richiamare altri *workflow* con l'*activity InvokeWorkflow* etc., rendono questo strumento estremamente flessibile e versatile per ogni tipo di esigenza. Per quanto la tecnologia non sia ancora matura si intravede già quali saranno i possibili sviluppi per il futuro, ovvero costruire ambienti che semplifichino ulteriormente la vita al programmatore "agganciandosi" a quella che è oggi la programmazione ad oggetti, sistemi che gestiscano anche la logica di un'applicazione oltre che la pura realizzazione.
Buon workflow.

Michele Locuratolo

CONTATTA L'AUTORE

L'autore può essere contattato attraverso il suo blog all'indirizzo http://blogs.mindbox.it/mighell e sarà lieto di rispondere ad ogni domanda.

CREARE APPLICAZIONI SERVICE ORIENTED

COMUNICAZIONE ED INTEROPERABILITÀ SONO LE PAROLE CHIAVE DEL FUTURO. VEDIAMO UNA DELLE SOLUZIONI MICROSOFT: PER LA CREAZIONE DI SOFTWARE CHE FA LARGO USO DI SISTEMI DISTRIBUITI: WINDOWS COMMUNICATION FOUNDATION

COM+, CORBA, Web Services, MSMQ sono alcuni dei sistemi che hanno dominato e attualmente dominano la scena della *Service Oriented Architecture* (SOA). L'unione delle loro caratteristiche consente di rispondere a molte delle necessità attuali, ma effettivamente non a tutte e non in maniera lineare. Durante gli anni ogni singola tecnologia è stata pensata e realizzata al nascere della relativa esigenza, dando vita ad un miscuglio di tecniche ognuna di esse con il proprio paradigma di programmazione in cui spesso è difficile districarsi. Oggi, l'esigenza di unificare e semplificare questa forma di sviluppo è stata la molla che ha dato vita a *Windows Communication Foundation* (a.k.a. Indigo).

PERCHÉ WINDOWS COMMUNICATION FOUNDATION

Connettività, applicazioni distribuite e interoperabilità sono i concetti alla base di WCF, un robusto framework che consente di creare complesse architetture *Service Oriented* con la semplicità con cui fino ad ora era possibile scrivere gli *XML Web Service.* Ma quali sono i vantaggi? Perché scegliere il framework WCF? Fino ad oggi la via intrapresa per lo sviluppo di una logica aziendale era prevalentemente costituita dalla scrittura di programmi *Object Oriented,* spesso però insufficiente per garantire una risposta adeguata alle diverse, crescenti esigenze.

In scenari in cui le necessità indirizzano lo sviluppo verso la realizzazione di applicazioni distribuite, l'iterazione tra i diversi blocchi di business layer in forma di servizi garantisce un approccio decisamente migliore. Come già accennato, i diversi sistemi attualmente disponibili sono già in grado di fornire un supporto allo sviluppo di servizi, ma costituiscono una selva oscura in cui il programmatore deve districarsi con non poche difficoltà prima di uscirne.

In **Tabella 1** possiamo vedere come WCF rappresenta la risposta alle diverse esigenze legate allo sviluppo *Service Oriented.*

Forse la tabella può far sembrare WCF un framework chiuso, non aperto verso applicazioni non Microsoft. L'affermazione è semplicemente sbagliata. Qualsiasi tecnologia in grado di consumare web services è anche capace di interagire con applicazioni scritte con WCF, questo perché il protocollo alla base delle comunicazioni è SOAP.

REQUISITI

Conoscenze richieste
Conoscenze base di programmazione in C#

Software
.NET Framework 2.0, WinFX beta 1

Impegno

Tempo di realizzazione

	ASMX	.NET Remoting	Enterprise Services	Web Service Extensions	MSMQ	Indigo
Web Services	X					X
Comunicazione .NET - .NET		X				X
Transazioni Distribuite			X			X
Supporto alle specifiche WS-*				X		X
Accodamento dei messaggi					X	X

Tabella 1: aaa

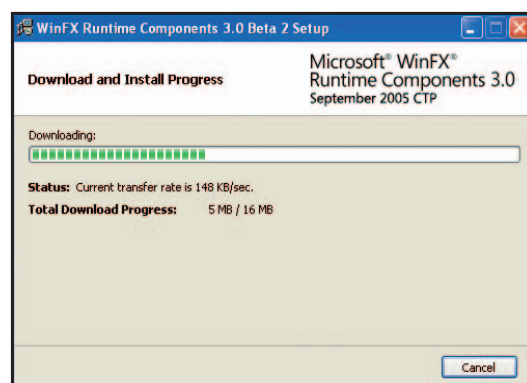



Fig. 1: ***Il setup di WinFX***

IL PRIMO SERVIZIO WINDOWS COMMUNICATION FOUNDATION

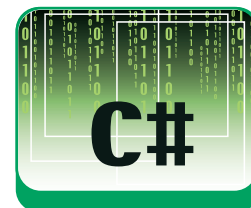
WCF, che al momento della stesura di questo articolo è ancora in versione beta, è basato sul .NET Framework versione 2.0 ed integrato in Visual Studio 2005. Attualmente, in riferimento a ciò che erano e

sono i Web Services, è possibile ospitare un servizio solo su un web server come IIS o Apache. In .NET 1.0 e 1.1, è tuttora possibile ottenere una adeguata comunicazione interprocesso attraverso l'utilizzo di *Remoting*, una tecnologia che consente di creare un server che espone dei servizi consumabili da client in esecuzione su application domain o processi, consentendo l'esecuzione server anche al di fuori di IIS. Ma se le tecnologie esistono, perché WCF? La possibilità di poter usufruire di un framework unico, flessibile ed estendibile, la possibilità di scindere il servizio dall'host che lo eseguirà, il tutto seguendo pochi semplici passi è la risposta alla domanda fatta. Come vedremo, in WCF un servizio può essere ospitato in diversi host:

- **Hosting o self-hosting in applicazioni** *(Console, Windows Forms o Windows Service)*
- **Hosting in IIS**.

In questo articolo proveremo ad utilizzare un servizio, in hosting su un'applicazione console ed utilizzato da un'altra applicazione console. Vedremo come, in maniera molto semplice, le due applicazioni comunicano tra di loro e come è possibile esporre lo stesso servizio anche come web service. Vediamo un po' di codice: dopo aver installato *WinFx*, il pacchetto contenente *Windows Communication Foundation*, per realizzare il servizio dobbiamo scrivere la nostra classe di servizio. Creiamo il file *RateService.cs* e aggiungiamo il seguente codice:

```
using System;
using System.ServiceModel;
namespace ioProgrammo.WCF
{
 [ServiceContract()]
 public interface IRateService
 {
     [OperationContract()]
     decimal CalcolaRata(decimal importo, int mesi);}
 [ServiceBehavior()]
 public class RateService : IRateService
 {
     [OperationBehavior()]
     public decimal CalcolaRata(decimal importo, int mesi)
     {
         decimal rata = (importo / mesi);
         Console.WriteLine("Importo: {0}", importo);
         Console.WriteLine("Mesi: {0}", mesi);
         Console.WriteLine("Rata: {0}", rata);
         return rata; } }
}
```

Cerchiamo esattamente di capire cosa rappresenta questo codice soffermandoci sulla firma dei metodi e sugli attributi utilizzati. Con WCF il presupposto allo sviluppo dei servizi è la definizione dei contratti. Un contratto stabilisce le operazioni che il servizio espone. Questo viene realizzato mediante la creazione di interfacce e classi marcate con attributi che ne stabiliscono il comportamento. Nel nostro esempio notiamo l'utilizzo dell'attributo *ServiceContract* per la definizione dell'interfaccia e dell'attributo *OperationContract* per la definizione dei metodi esposti. La classe che si occupa di implementare il contratto definito nell'interfaccia, viene invece marcata con gli attributi *ServiceBehavior* e *OperationBehavior*. Questa nuova definizione sostituisce quella finora utilizzata per i tradizionali Xml Web Services, dove l'attributo WebService definisce le caratteristiche della classe di servizi, mentre l'attributo *WebMethod* rappresenta i singoli metodi (o servizi) esposti e le rispettive impostazioni.

I QUATTRO PILASTRI DELL'ARCHITETTURA SERVICE-ORIENTED

Le architetture service-oriented definiscono quattro concetti detti pilastri:

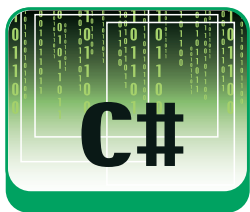
1. **Definizione di confini espliciti: i limiti entro i quali un servizio deve muoversi e funzionare devono essere ben definiti. È importante distinguere l'accesso ad oggetti locali dall'accesso ad un servizio.**
2. **I servizi sono autonomi: ogni servizio deve essere capace di funzionare senza dipendere da altri servizi.**
3. **Un servizio condivide schemi e contratti e non classi ed interfacce: ogni servizio accetta uno schema di dati, costituendo così architetture tra loro disaccoppiate che rimangono stabili nel tempo.**
4. **Compatibilità dei servizi basata su policy: ogni servizio deve stabilire esplicite regole per il suo utilizzo in forma leggibile, in modo da consentire la separazione tra il servizio e le modalità di accesso al servizio.**

UNA CONSOLE COME HOSTING

Dopo aver definito il nostro servizio, creiamo ora l'hosting che deve ospitarne l'esecuzione. Creiamo un nuovo file *Server.cs* ed inseriamo il seguente codice:

```
using System;
using System.ServiceModel;
namespace ioProgrammo.WCF
{
 public class Server
 {
  public static void Main(string[] args)
  {
   using (ServiceHost host = new ServiceHost(typeof(
                         ioProgrammo.WCF.RateService)))
   {
    Uri address = new Uri(
                  "http://localhost:8000/RateService/");
    host.AddEndpoint(
                typeof(ioProgrammo.WCF.IRateService),
```

SOAP (SIMPLE OBJECT ACCESS PROTOCOL)
Protocollo per lo scambio di dati tra diverse applicazioni.

ENDPOINT
Indica uno specifico punto di accesso di un Web Service utilizzando un tipo di dati e un determinato protocollo.

```
                        new WSHttpBinding(), address);
    host.Open();
    Console.WriteLine("Service Started.");
    Console.ReadLine();
    Console.WriteLine("Service Stopped!!!");
    host.Close(); } } }
}
```

Come potete notare gran parte del lavoro è svolto dalla classe *ServiceHost* che, dopo aver dichiarato l'endpoint, apre la connessione e si mette in attesa di chiamate in ingresso. Il funzionamento dell'host ha luogo mediante la creazione di uno o più endpoints ognuno dei quali imposta un contratto da utilizzare, la modalità di comunicazione con il servizio e un indirizzo che indica dove è possibile contattare l'endpoint. Nell'esempio il contratto è costituito dall'interfaccia *IRateService* marcata con l'attributo *ServiceContract,* il tipo di comunicazione utilizzata dall'*endpoint,* definita dall'oggetto *WSHttpBinding,* ed infine l'indirizzo da raggiungere per poter eseguire il servizio.

L'ABC DI WINDOWS COMMUNICATION FOUNDATION

I tre elementi chiave di un *endpoint* che chiariscono il suo significato costituiscono l'*ABC* di *WCF*:

- **Address: indica l'indirizzo del servizio sotto forma di *Uri*. Esempio: http://localhost:8000/MyService oppure soap://localhost:8000/MyService**
- **Binding: stabilisce le modalità di comunicazione con il servizio. Esso è l'insieme delle informazioni che controllano il trasporto, il protocollo e la codifica utilizzata per comunicare con l'*endpoint.***
- **Contract: è la definizione di ciò che il servizio fa, specificandone le operazioni e i messaggi.**

Tutti i tre elementi sono presenti nel WSDL nei tag wsdl:service, *wsdl:binding, wsdl:portType* e *wsdl:message.*

BIBLIOGRAFIA

- **PROGRAMMING INDIGO** ***David Pallmann*** **(Microsoft Press) ISBN 0-7356-2151-9 2005**
- **WEB SERVICES ARCHITECTURE AND ITS SPECIFICATIONS: ESSENTIALS FOR UNDERSTANDING WS-*** ***Luis Felipe Cabrera Chris Kurt*** **(Microsoft Press) ISBN 0-7356-2162-4 2005**

Nell'esempio viene utilizzato l'http come protocollo di trasporto, ma WCF consente l'uso anche di altri protocolli come *tcp* o *udp,* modificando semplicemente il prefisso dell'uri ed il relativo tipo di binding. Se, ad esempio, volessimo utilizzare un canale tcp, è sufficiente sostituire *net.tcp://localhost:8000/RateService/* all'indirizzo indicato ed impostare il tipo di binding attraverso l'oggetto *NetTcpBinding.* È anche possibile creare dei binding personalizzati attraverso l'oggetto *CustomBinding.* Nel nostro esempio, l'utilizzo di *WSHttpBinding* garantisce l'affidabilità della comunicazione *end-to-end,* risultando però inutilizzabile per le applicazioni che necessitano delle funzioni di accodamento dei messaggi. Proviamo ora a formulare l'ipotesi di un client che deve utilizzare il nostro servizio. Creiamo un nuovo file *Client.cs* ed aggiungiamo il seguente codice:

```
using System;
using System.ServiceModel;
namespace ioProgrammo.WCF
{
 public class Client
  {
   public static void Main(string[] args)
    {
     Uri address = new Uri(
                 "http://localhost:8000/ServerCalculator/");
     ioProgrammo.WCF.IServerCalculator proxy = null;
     proxy = ChannelFactory.CreateChannel
                    <ioProgrammo.WCF.IServerCalculator>
                           (address, new WSHttpBinding());
     decimal importo, importoRata;
     int mesi;
     Console.Write("Importo: ");
     importo = decimal.Parse(Console.ReadLine());
     Console.Write("Mesi: ");
     mesi = int.Parse(Console.ReadLine());
     importoRata = proxy.CalcolaRata(importo, mesi);
     Console.WriteLine("ImportoRata: {0}", importoRata);
    } }
}
```

Così come nell'applicazione host, anche nel client è necessario definire l'endpoint da raggiungere. Perciò nel codice riportato creiamo un canale passando l'indirizzo e il tipo di comunicazione da attivare, ed utilizziamo le potenzialità dei generics per ottenere l'istanza remota del servizio. Infine viene effettuata una semplice chiamata al metodo *CalcolaRata.*
In **Figura 2** potete vedere il risultato della comunicazione tra l'applicazione client e l'applicazione server.

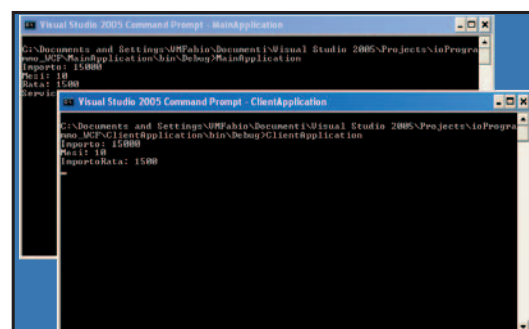



Fig. 2: ***La comunicazione tra client e server***

SERVE UN HOSTING? ECCO IIS…

Lo stesso servizio, però, può essere esposto anche come web services e quindi ospitato sul web server IIS. A questo scopo, WCF introduce la nuova estensione *.svc* per identificare un web service e distinguerlo dai classici *.aspx.* La configurazione in questo caso viene gestita tutta attraverso il file *web.config.* Creiamo una nuova directory virtuale su IIS ed inseriamo nella relativa cartella un file

web.config con il seguente codice:

```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com
                    /.NetConfiguration/v2.0">
  <system.serviceModel>
    <services>
      <service
        type="ioProgrammo.WCF.ServerCalculator">
        <endpoint
          address=""
          binding="wsHttpBinding"
          bindingConfiguration="Binding1"
          contract="ioProgrammo
                  .WCF.IServerCalculator" />
      </service>
    </services>
    <bindings>
      <wsHttpBinding>
        <binding
          configurationName="Binding1"/>
      </wsHttpBinding>
    </bindings>
  </system.serviceModel>
</configuration>
```

Notate che tutto quello fatto nell'applicazione console eseguita come hosting del servizio, viene esattamente replicato nel *web.config*. Abbiamo semplicemente omesso l'attributo address del tag *endpoint* perché viene utilizzato l'indirizzo esposto da IIS.
Dopo la creazione del web.config posizioniamo nella cartella *bin* della directory virtuale l'assembly contenente il servizio ed infine scriviamo la pagina che si occuperà di esporre il nostro web service, la chiamiamo *WSCalculator.svc* e ci inseriamo il seguente codice:

```
<%@Service language=C# Debug="True" class=
           "ioProgrammo.WCF.ServerCalculator" %>
<%@Assembly Name="assemblyName" %>
```

Chi ha lavorato con ASP.NET anche nelle versioni precedenti noterà una certa familiarità nella sintassi. Infatti la prima direttiva *@Service* dichiara il tipo di pagina come servizio ed indica la classe che ne espone le funzionalità. La seconda direttiva *@Assembly* viene utilizzata per indicare al compilatore *JIT* di ASP.NET in che assembly deve recuperare la classe di servizio. La direttiva *@Assembly* espone anche l'attributo src utile per referenziare un file di codice invece che un assembly. A questo punto se proviamo a digitare l'indirizzo del servizio nel browser avremo un risultato simile a quello di **Figura 3**.
Ora possiamo generare la classe *proxy* che esporrà ai client le funzionalità *Web Service WCF*. L'uso del tool *svcutil.exe* dal prompt dei comandi semplifica notevolmente questo passaggio:

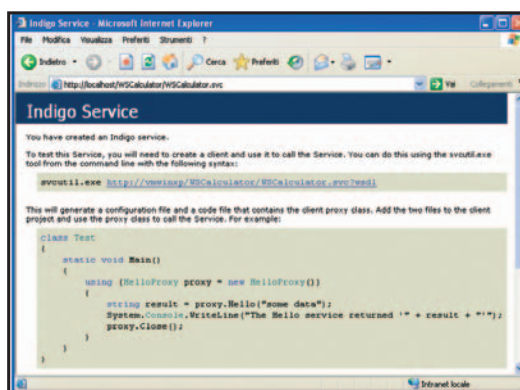



Fig. 3: **La home page del servizio**

```
svcutil.exe http://localhost/WSCalculator
                        /WSCalculator.svc?wsdl
```

come anche indicato nella pagina di benvenuto del web service. Un qualsiasi applicazione client scritta con il framework 2.0 non deve far altro che includere ed utilizzare la classe *proxy* per poter consumare il web service.

CONCLUSIONI

Nell'articolo abbiamo fatto una semplice panoramica di ciò che *Windows Comunication Foundation* può offrire, concentrandoci sulla versatilità del nuovo framework pensato per lo sviluppo di architetture Service-Oriented. Abbiamo visto come è semplice realizzare un servizio ed esporlo attraverso diversi tipi di hosting, ma anche come è semplice includerlo ed utilizzarlo nelle applicazioni client. Sono personalmente molto affascinato da questa tecnologia e spero di aver stuzzicato la vostra fantasia. L'articolo è basato su una versione beta di WCF, perciò è possibile che qualcosa cambi, come è già avvenuto, nel corso delle successive release, perciò vi invito a contattarmi per chiarimenti, approfondimenti o critiche attraverso il forum o i contatti presenti nel box laterale.
Buon lavoro.

Fabio Cozzolino

Microsoft Windows Communication Foundation
http://msdn.microsoft.com/webservices/indigo

WinFX Runtime Components
http://www.microsoft.com/downloads/details.aspx?familyid=FFD636F0-86E9-41E8-9E1C-100A4CC4888F&displaylang=en

WinFX SDK
http://www.microsoft.com/downloads/details.aspx?familyid=C20CC6C8-1F4E-4A5C-BC79-C2FE9ABE69AA&displaylang=en

Il mio blog
http://blogs.ugidotnet.org/fabioc

COMPILARE I SORGENTI DELL'ARTICOLO

Per compilare la classe di servizio utilizziamo la seguente istruzione a riga di comando:

```
csc RateService.cs /target:library
 /out:RateService.dll/reference:System,
                     System.ServiceModel
```

Per l'applicazione console server scriviamo:

```
csc /target:exe /out:MainApplication
      Server.cs /reference:System.dll,
                  System.ServiceModel.dll,
                        RateService.dll
```

Per l'applicazione console client compiliamo con questa istruzione:

```
csc /target:exe /out:ClientApplication
        Client.cs/reference:System.dll,
                System.ServiceModel.dll,
                        RateService.dll
```

Il servizio esposto tramite IIS viene compilato "Jus in Time".

PROCESSI ASINCRONI CON FACILITA'

FACCIAMO CONOSCENZA CON "BACKGROUNDWORKER" UN COMPONENTE PRESENTE IN VISUAL STUDIO 2005, CHE SEMPLIFICA ENORMEMENTE LA VITA AL PROGRAMMATORE METTENDOGLI A DISPOSIZIONE AUTOMATISMI PER LA GESTIONE DEI THREAD

La naturale evoluzione dell'informatica ci ha condotto a poter usare PC sempre più potenti, con sempre più memoria e con hard-disk più capienti. Tutto questo consente ai computer di poter compiere operazioni impensabili fino a qualche tempo fa. La velocità dei microprocessori attuali permette di elaborare grandi quantità di dati, immagini, video o grandi database. Nonostante questa eccezionale dotazione in termini di hardware, il 90% delle performance delle applicazioni, rimane nelle mani di noi programmatori. Ma cosa succederebbe se il nostro programma mentre sta compiendo un'elaborazione lunghissima su un'immagine bloccasse ogni altra operazione, in attesa che la prima termini? Molto probabilmente la nostra applicazione andrebbe in stallo fino a quando l'elaborazione non fosse completa. Questo modo di procedere è largamente utilizzato, noto come utilizzo di un processo sincrono, nel quale l'utente deve obbligatoriamente aspettare la fine dell'operazione in corso prima di poter fare qualcosa d'altro. Non è detto che non si possano utilizzare processi sincroni nello sviluppo, ma questa soluzione è ottimale solo nel caso di operazioni veloci, oppure quando è proprio necessario aspettare la fine di un'operazione prima di proseguire con la successiva. Considerare il caso in cui vogliate salvare un file sull'hard disk ed inviare un file via e-mail, è necessario che la scrittura sia completata prima di procedere alla spedizione. In caso di operazioni che possono essere svolte senza dipendenze dalle precedenti è invece utile implementare soluzioni asincrone in cui più task possono essere svolti in parallelo. Pensiamo ad esempio alla masterizzazione di un DVD o alla scansione di un'immagine. Questi sono tipici casi di processi asincroni, cioè porzioni di codice che girano parallelamente ad altri processi in esecuzione. Il nostro codice fa qualcosa senza che l'utente debba per forza aspettarne la fine: l'importante è poter disporre di uno strumento per verificare lo stato dell'elaborazione e per poter ricevere un avviso quando l'elaborazione è terminata. Lo scopo di questo articolo sarà proprio illustrare come programmare task asincroni all'interno di una nostra applicazione, tramite alcune classi di .NET, fra le quali spicca in modo particolare il componente *BackgroundWorker.*

UN PRIMO APPROCCIO

Supponiamo di lavorare ad un'applicazione che, data una fotografia ad alta risoluzione, ne generi il negativo. Se programmassimo il tutto in modo sincrono, durante l'elaborazione dell'immagine, fatta pixel per pixel, l'interfaccia rimarrebbe bloccata, e non potremo spostare la Windows Forms, non potremo ridimensionarla, non potremo interagire in alcun modo con il sotware. L'applicazione dovrà aspettare la conclusione dell'elaborazione prima di procedere con la prossima istruzione. Molti linguaggi di programmazione, propongono costrutti per la creazione di thread secondari, lasciando nelle mani del programmatore il compito di occuparsi anche delle procedure di più basso livello. Anche .NET consente la creazione di thread a basso livello, ma contemporaneamente mette a disposizione il componente *BackgroundWorker* che "nasconde" al programmatore le problematiche di più basso livello, mettendogli a disposizione una serie di opzioni ad alto livello che consentono di gestire i *thread* in un maniera estremamente efficace ma molto semplice. Usando il *BackgroundWorker*, creiamo a tutti gli effetti un thread secondario che scansiona l'immagine fino alla fine, lasciando però l'applicazione attiva e funzionante. L'utente può quindi aprire menu, attivare altre funzioni, ovviamente facendo attenzione (ma questo è compito nostro di noi sviluppatori) che il tutto non vada in conflitto con il *thread* gestito dal *BackgroundWorker*. Per poter funzionare correttamente, il *BackgroundWorker* deve conoscere sostanzialmente tre cose: cosa deve eseguire, cosa notificare all'utente durante il processo e come notificarglielo, cosa deve fare al termine del processo. Tecnicamente, il tutto è implementato attraverso tre eventi esposti dalla classe: *DoWork, ProgressChanged* e *RunWorkerCompleted.*

IN PRATICA

La nostra applicazione farà esattamente quanto fin qui descritto, ovvero elaborare un'immagine, e avviare questa elaborazione in un thread secondario di modo che l'applicazione rimanga disponibile per ulteriori operazioni. In fase di sviluppo, la nostra form sarà composta da

- una PictureBox (*picOriginale*): conterrà l'immagine originale così come caricata dall'hard disk;
- una PictureBox (*picElaborata*): conterrà l'immagine modificata a seconda dell'algoritmo che abbiamo deciso di applicare;
- un Button (*btnSfoglia*): userà la classe *OpenFileDialog* per scegliere ed aprire quale immagine vogliamo elaborare;
- un Button (*btnNegativo*): cliccando, l'immagine in *picOriginale* verrà resa negativa e ridisegnata in *picElaborata;*
- un Button (*btnAnnulla*): cliccandolo, l'utente può abortire l'esecuzione del processo asincrono;
- un BackgroundWorker (*wrkElabora*): ci servirà per eseguire processi asincroni di elaborazione sull'immagine.

Il bottone ci servirà per selezionare l'immagine da elaborare, un doppio clic sulla voce di menu avvierà invece il processo di elaborazione, nella stessa applicazione ci saranno poi previste altre opzioni, ma per iniziare è sufficiente avere conoscenza semplicemente di questa prima implementazione. Ovviamente nella stessa form avremo anche un *BackgroundWorker* che ci servirà per gestire il thread, e una progressbar che utilizzeremo per visualizzare lo stato d'avanzamento dell'elaborazione.

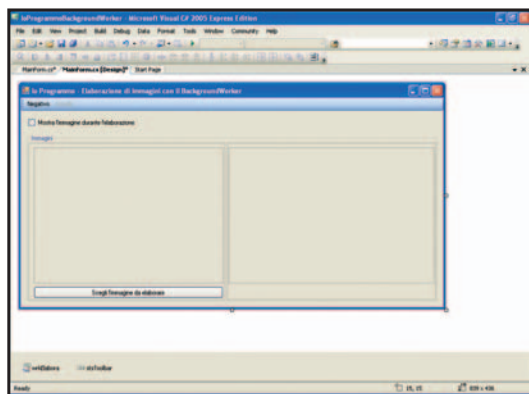



Fig. 1: ***Ecco come sarà composta la nostra form in fase di elaborazione***

La classe *BackgroundWorker* è un componente: questo significa che quando lo si trascina dalla toolbox di Visual Studio 2005 sulla Windows Forms, esso verrà posizionato nella tray-area. Di conseguenza, rimarrà invisibile a run-time, mentre durante lo sviluppo *(design-time)* lo possiamo semplicemente selezionare dalla tray-area, nella parte inferiore della Windows Forms. In *design-time*, il *BackgroundWorker* non offre molte possibilità. Le uniche proprietà disponibili sono due: *WorkerReportsProgress* e *WorkerSupportsCancellation.* Entrambe sono proprietà booleane: la prima attiva la possibilità di notificare all'utente lo stato in cui si trova il processo asincrono (ad esempio con una *ProgressBar,* o un'animazione). La seconda proprietà, attiva o meno la possibilità da parte dell'utente di annullare l'esecuzione del processo asincrono. Dopo aver posizionato il *BackgroundWorker,* vediamo di gestire i tre eventi accennati prima: *DoWork, ProgressChanged* e *RunWorkerCompleted.* Supponendo quindi di avere un *BackgroundWorker* chiamato *wrkElabora,* cominciamo con il ciccare due volte sul bottone, per gestire il codice che selezionerà l'immagine da elaborare. Il nostro codice avrà la seguente forma:

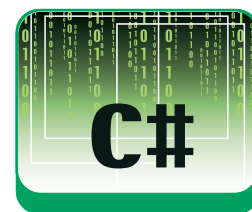
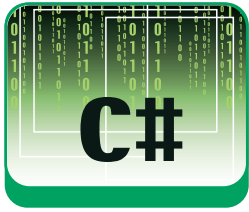
```
private void btnSfoglia_Click(object sender, EventArgs e)
{
  /* Sfoglio un'immagine e la salvo nella PictureBox
  * picOriginale */
  OpenFileDialog dlg = new OpenFileDialog();
  dlg.AddExtension = true;
  dlg.CheckFileExists = true;
  dlg.CheckPathExists = true;
  dlg.Multiselect = false;
  dlg.RestoreDirectory = true;
  DialogResult res = dlg.ShowDialog();
  if (res == DialogResult.OK)
  {
     Image img = Image.FromFile(dlg.FileName);
     this.picOriginale.Image = img;
  }
  dlg.Dispose();
}
```

a questo punto disponiamo dell'immagine originale nel controllo *picOriginale.* Iniziamo l'elaborazione cliccando due volte sul menu "Negativo". Il codice di gestione è il seguente:

```
  private void btnNegativo_Click(object sender,
```

LE PROPRIETÀ PIÙ UTILI PER GESTIRE MEGLIO IL BACKGROUNDWORKER

Abbiamo già visto le proprietà più importanti esposte del componente *BackgroundWorker.*

- **CancellationPending - ritorna un bool che ci dice se l'utente ha richiesto oppure no l'interruzione del task asincrono.**
- **IsBusy - essenziale per evitare di cominciare un nuovo task se ne è già in esecuzione un altro: il framework in questo caso genera un'eccezione *InvalidOperationException* che va gestita come di consueto.**
- **WorkerReports Progress - attiva o meno la possibilità di aggiornare l'interfaccia utente della nostra applicazione durante l'esecuzione.**
- **WorkerSupports Cancellation - infine, attiva o meno la possibilità di annullare l'esecuzione da parte dell'utente.**

```
                                        EventArgs e)
{
    // esco se qualcosa non è ok
    if (this.picOriginale.Image == null) return;
    if (wrkElabora.IsBusy) return;
    // setto gli handler ed imposto canceled = false
    cancelled = false;
    this.btnAnnulla.Enabled = !wrkElabora.IsBusy;
    this.btnNegativo.Enabled = !wrkElabora.IsBusy;
    this.wrkElabora = new BackgroundWorker();
    this.wrkElabora.WorkerReportsProgress = true;
    this.wrkElabora.WorkerSupportsCancellation = true;
    this.wrkElabora.DoWork += new
                DoWorkEventHandler(Negativo_DoWork);
    this.wrkElabora.ProgressChanged += new
                        ProgressChangedEventHandler(
                          Negativo_ProgressChanged);
    this.wrkElabora.RunWorkerCompleted += new
                     RunWorkerCompletedEventHandler(
                        Negativo_RunWorkerCompleted);
    this.wrkElabora.RunWorkerAsync();
}
```

Concentriamoci sulle linee di codice

```
if (wrkElabora.IsBusy) return;
this.wrkElabora.DoWork += new
                DoWorkEventHandler(Negativo_DoWork);
this.wrkElabora.RunWorkerAsync();
```

Essenzialmente, il *BackgroundWorker* vuole sapere cosa deve eseguire in modo asincrono: questo risultato lo si ottiene associando un opportuno handler all'evento *DoWork*. Nel nostro caso, l'handler è *Negativo_DoWork*, per cui ci basta implementare una funzione, il cui codice può ad esempio essere il seguente:

```
void Negativo_DoWork(object sender,
                                DoWorkEventArgs e)
{
    inizio = DateTime.Now;
    Color clrOrig;
    Color clrElab;
    bmpOrig = new Bitmap(this.picOriginale.Image);
    bmpElab = new Bitmap(this.picOriginale.Image);
    int width = bmpOrig.Width;
    int height = bmpOrig.Height;
    pixelTotali = width * height;
    // elaboro, uno ad uno, tutti i pixels dell'immagine
    for (y = 0; y < height; y++)
    {
        for (x = 0; x < width; x++)
        {
            clrOrig = bmpOrig.GetPixel(x, y);
            clrElab = Color.FromArgb(
                255 - clrOrig.R,
                255 - clrOrig.G,
                255 - clrOrig.B);
            bmpElab.SetPixel(x, y, clrElab);
        }
    }
}
```

Il codice non fa altro che cambiare il colore di ogni pixel, rendendo l'intera immagine negativa. Nello scrivere codice di questo tipo - che gira in background - dobbiamo prestare attenzione all'accesso delle risorse: trattandosi di un processo asincrono infatti può succedere che un'altra applicazione tenti di accedere alla stessa informazione nello stesso istante. Inoltre, può accadere che il *BackgroundWorker* crei due thread diversi che tentano di scrivere sullo stesso file, oppure di accedere allo stesso oggetto sulla Windows Forms, e così via. Di fronte a questa problematica, possiamo comportarci in modi diversi: possiamo controllare la proprietà *IsBusy*, che ci dice se il *BackgroundWorker* sta già eseguendo qualcosa. Oppure, ed è la soluzione più performante e sicura, possiamo utilizzare la keyword lock di C# che consente di marcare una porzione di codice come critica, assicurando che un determinato oggetto sia utilizzato da un solo thread alla volta. Possiamo quindi decidere di implementare nel ciclo for più interno del codice sopra come indicato di seguito:

```
lock(bmpElab)
{
    bmpElab.SetPixel(x, y, clrElab);
}
```

Nel nostro esempio, l'oggetto critico è la Bitmap *bmpElab*, che contiene l'immagine modificata: di conseguenza, inseriamo questo codice in un blocco *lock*, che ci assicura che quell'oggetto in quel dato istante venga manipolata da un solo *thread*.

COMUNICAZIONI ALL'UTENTE

In condizioni normali, in .NET dobbiamo fare attenzione ad aggiornare l'interfaccia utente, perché possiamo farlo esclusivamente dal thread principale della nostra applicazione. Il componente *BackgroundWorker* nasconde questa particolare complessità. Di conseguenza, l'unica cosa richiesta è l'implementazione di due handler diversi: uno per l'evento *ProgressChanged* ed uno per l'evento *RunWorkerCompleted*. Come è facile intuire, il primo viene sollevato durante l'esecuzione - a nostra discrezione - mentre il secondo avviene al termine dell'esecuzione, oppure quando l'esecuzione viene annullata. Gli handler vengono associati come di consueto:

```
this.wrkElabora.ProgressChanged +=
                    new ProgressChangedEventHandler(
                        Negativo_ProgressChanged);
this.wrkElabora.RunWorkerCompleted += new
                    RunWorkerCompletedEventHandler(
                        Negativo_RunWorkerCompleted);
```

Per poter gestire correttamente l'evento *ProgressChanged*, è necessario impostare a true la proprietà *WorkerReportsProgress*. All'interno della funzione *Negativo_DoWork* che abbiamo implementato prima, quindi, solleviamo l'evento *ProgressChanged*:

```
Negativo_ProgressChanged (this, new
            ProgressChangedEventArgs(0, bmpElab));
```

che viene gestito dal suo corrispondente handler.

```
void Negativo_ProgressChanged(object sender,
                        ProgressChangedEventArgs e)
{
    this.picElaborata.Image = (Bitmap)e.UserState;
}
```

In pratica, l'utente può sapere a che punto è l'elaborazione dell'immagine. L'handler utilizza la classe *ProgressChangedEventArgs*, derivata da *EventArgs*, che permette di passare parametri dal thread che regola il processo asincrono. Infine, non ci resta che gestire l'evento *RunWorkerCompleted*, nel quale avvisiamo l'utente che l'operazione è terminata.

```
void Negativo_RunWorkerCompleted(object sender,
                    RunWorkerCompletedEventArgs e)
{
    string output;

    /* Scrivo l'immagine modificata sulla PictureBox
    * picElaborata */
    this.picElaborata.Image = bmpElab;

   output = string.Format("Operazione
            COMPLETATA!\nSono stati modificati {0}
                    pixels!\n\nTempo : {1}.", x * y,
                            intervallo.ToString());
    MessageBox.Show(output, "IoProgrammo");
}
```

Nel codice qui sopra, renderizzo su *picElaborata* l'immagine in negativo, e poi avviso l'utente con una normale *MessageBox*.

ANNULLARE L'ESECUZIONE ASINCRONA

Una delle possibilità più interessanti che possiamo offrire all'utente è la possibilità di interrompere l'esecuzione del processo, nel caso in cui voglia tornare indietro o ad esempio ripeterla con parametri differenti. Il *BackgroundWorker* ovviamente consente questa tecnica, nella quale dobbiamo però gestire le cose manualmente.
Per prima cosa, dobbiamo impostare a *true* la proprietà *WorkerSupportsCancellation* del *BackgroundWorker*. Successivamente, dobbiamo per esempio aggiungere un nuovo Button sulla Windows Form, che l'utente può cliccare per richiedere l'interruzione.
Possiamo poi aggiungere un Button *btnAnnulla* sulla form, associando il seguente codice:

```
private void btnAnnulla_Click(object sender,
                                        EventArgs e)
{
    this.wrkElabora.CancelAsync();
}
```

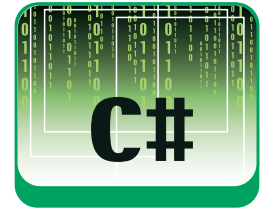
L'interruzione del processo asincrono non è completamente automatica. La chiamata al metodo *CancelAsync* non fa altro che impostare la proprietà *CancellationPending* del componente, che dobbiamo quindi monitorare nella function *DoWork* per bloccare a tutti gli effetti.

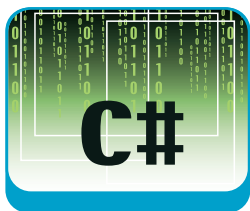
```
if (wrkElabora.CancellationPending)
{
    // forzo l'uscita dal loop
    break;
}
```

Il blocco if riportato qui sopra controlla la proprietà *CancellationPending:* se vale true, il loop contenuto nella funzione *Negativo_DoWork* viene interrotto. Non dimentichiamoci che anche se l'utente interrompe l'esecuzione del processo, viene comunque scatenato l'evento *RunWorkerCompleted.* Spetta al nostro codice discriminare le modalità con cui l'esecuzione del processo termina: il codice C# completo presente nel progetto allegato utilizza ad esempio una variabile booleana.

CONCLUSIONI

Il *BackgroundWorker* è un efficace componente del Framework 2.0, semplice da usare e veloce da integrare nel nostro codice, che con poco sforzo ci permette di realizzare software di una certa complessità, che consentono di eseguire processi in background, mentre l'utente continua ad utilizzare la nostra applicazione.
Il *BackgroundWorker* nasconde allo sviluppatore molta della complessità presente invece in altri componenti di altri linguaggi di programmazione.

Igor Damiani

IMPARIAMO A USARE LE STORED PROCEDURES

ECCO COME POSSIAMO ACCELERARE LE PRESTAZIONI DEL DATABASE MEMORIZZANDO LE QUERY DIRETTAMENTE SUL SERVER, E LASCIANDO AL DATABASE IL COMPITO DI GESTIRLE AL MEGLIO

Problema: stiamo scrivendo il gestionale della nostra vita, quello che ci lancerà nell'universo dei programmatori ricchi e famosi! All'interno di questo gestionale utilizziamo frequentemente una query chilometrica. Tutte le volte che ci occorre la nostra query, siamo costretti a copiarla da una parte all'altra del codice. Siccome siamo bravi, abbiamo creato una classe per gestire la query in questione di modo che, se volessimo cambiarla, dovremmo farlo solo una volta. Tuttavia, all'interno del programma, siamo costretti ogni volta ad instanziare un oggetto della classe ecc, inoltre la classe stessa al suo interno dovrà provvedere alla connessione con il database, l'invio dei comandi SQL ed ogni altra operazione che sia necessaria a gestire il tutto, con conseguente spreco di risorse e agilità del nostro software. Proviamo a fare un'altra ipotesi. E se la query sql fosse direttamente immagazzinata in sql server? Ad esempio potremmo salvare la nostra query in sql server ed attribuirli un nome, per poi richiamarla dall'interno del nostro programma direttamente tramite la sua etichetta. Quali vantaggi ci porterebbe questo approccio? Sicuramente il nostro programma sarebbe più leggero, la query sarebbe disponibile anche ad altri software che potrebbero utilizzarla in modo trasparente senza conoscerne realmente i dettagli implementativi, il risultato sarebbe leggermente più veloce in quanto eseguita dal db stesso senza ricorrere ad oggetti intermedi. Quello che faremo in questo articolo sarà esattamente imparare a creare query "immagazzinate" direttamente in SQL Server, ovvero quelle che in linguaggio tecnico vengono chiamate *"Stored Procedure"*. Per tutti i vostri esperimenti potrete usare Microsoft SQL server express edition 2005 allegato al CD presente in questo stesso numero di ioProgrammo.

SI INIZIA!

Tecnicamente una stored procedure è composta da una serie di istruzioni *T-SQL* memorizzate in SQL Server. L'insieme delle istruzioni preso come elemento unico rappresenta una *Stored Procedure,* che ovviamente viene identificata da un *nime.* A differenza delle query eseguite nel query analyzer, o via codice, le stored procedure vengono compilate da SQL Server prima di essere eseguite. È chiaro, quindi, che una stored procedure è tipicamente più veloce della query corrispondente. Il miglioramento delle prestazioni non è l'unico vantaggio delle stored procedure, isolamento e crittografia sono altri aspetti che non possiamo trascurare.
Senza entrare nel dettaglio, pensiamo alla possibilità di limitare i privilegi sulla struttura del database agli utenti delle nostre applicazioni consentendo l'accesso alla singola stored procedure che recupera i dati e non un accesso diretto alle tabelle. Esiste poi la possibilità di crittografare una stored procedure in modo da non permettere l'accesso alle istruzioni T-SQL che la compongono.
Le stored procedure, una volta memorizzate, possono essere eseguite sia nel query analyzer sia nei nostri programmi come fossero dei comandi T-SQL. Si evidenzia in questo modo un altro vantaggio derivante dal loro utilizzo: supponiamo di avere una query che eseguiamo in diverse applicazioni e di renderci conto dopo un periodo di utilizzo che questa può essere ottimizzata. In teoria dovremmo andare a modificare tutte le nostre applicazioni. Mentre se utilizzassimo una stored procedure sarebbe sufficiente modificarla all'interno del SQL Server per ottenere i benefici desiderati in tutte le applicazioni che la utilizzano.

CREARE UNA STORED PROCEDURE

Esistono due modi per creare una stored procedure: il primo passa attraverso tramite l'istruzione *CREATE PROCEDURE* nel query

analyzer, il secondo fa uso esclusivamente dell'*enterprise manager*. Gli esempi di questo articolo faranno riferimento ad una tabella chiamata *Utenti* formata dai campi: *idUtente, nome, cognome, email* e *dataScadenza*.
La query T-SQL per la creazione della tabella è la seguente:

```
CREATE TABLE [Utenti] (
    [idUtente] [int] IDENTITY (1, 1) NOT NULL ,
    [nome] [varchar] (50) COLLATE
                    Latin1_General_CI_AS NOT NULL ,
    [cognome] [varchar] (50) COLLATE
                    Latin1_General_CI_AS NOT NULL ,
    [email] [varchar] (50) COLLATE
                    Latin1_General_CI_AS NOT NULL ,
    [dataScadenza] [datetime] NOT NULL
             CONSTRAINT [DF_Utenti_dataIscrizione]
            DEFAULT (DATEADD(yyyy, 1, GETDATE())),
    CONSTRAINT [PK_Utenti] PRIMARY KEY  CLUSTERED
    (
      [idUtente]
    )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

Una volta creata la tabella, popolatela pure con dei dati, di modo che il resto dell'articolo sia maggiormente comprensibile.

LA PRIMA STORED PROCEDURE

Supponiamo di voler leggere nome ed email dalla tabella *Utenti* appena creata, ordinati per nome. La query SQL sarebbe molto semplice

```
SELECT nome, email
FROM Utenti
ORDER BY nome
```

Partendo da questa query creiamo la nostra prima stored procedure:

```
CREATE PROCEDURE dbo.listUsers
AS
SELECT nome, email
FROM Utenti
ORDER BY nome
```

Eseguendo il codice T-SQL, viene creata la stored procedure *"dbo.listUsers"* che, se eseguita, restituirà nome ed email degli utenti presenti nella tabella. Per eseguire la stored procedure è sufficiente digitarne il nome (il comando *EXEC* è facoltativo).

```
EXEC dbo.listUsers
```

Vediamo in figura il risultato dell'esecuzione

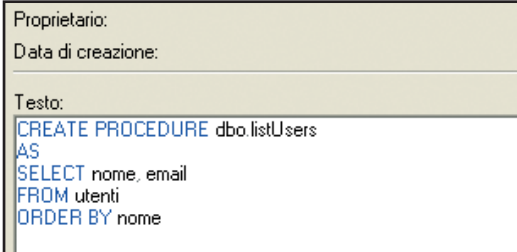
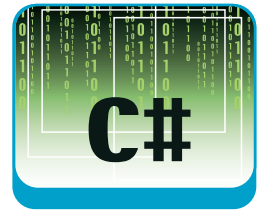



Fig. 2 **Definizione dello script T-SQL**

CREAZIONE DI STORED PROCEDURE DA ENTERPRISE MANAGER

Le stored procedure possono essere create anche utilizzando l'enterprise manager di SQL Server. Per farlo, clicchiamo con il tasto destro del mouse *"Stored procedure"* per il database che ci interessa, e selezioniamo *"Nuova stored procedure..."*.
Scriviamo quindi la nostra query T-SQL e clicchiamo su *Ok*.

CRIPTARE UNA STORED PROCEDURE

È possibile specificare l'opzione *WITH ENCRYPTION* per proteggere la query della nostra stored procedure.

```
CREATE PROCEDURE dbo.listUsersEnc WITH
                                   ENCRYPTION
AS
SELECT   nome,  email
FROM      Utenti
WHERE
DATEDIFF( d, GETDATE(), dataScadenza ) = 7
```

La stored procedure viene criptata e il contenuto T-SQL risulta protetto da visualizzazioni indesiderate.

MODIFICA E CANCELLAZIONE DI STORED PROCEDURE

Possiamo poi modificare o eliminare una stored procedure grazie alle istruzioni *ALTER* e *DROP*. Supponiamo di voler visualizzare solo gli utenti la cui iscrizione scade tra una settimana:

T-SQL

T-SQL, acronimo di Transact-SQL è la versione proprietaria di SQL utilizzata da Microsoft SQL Server. Ne mantiene i costrutti base ma fornisce funzionalità aggiuntive per la manipolazione e l'estrazione dei dati. È un vero e proprio linguaggio di programmazione che consente di utilizzare variabili, fornisce una serie di istruzioni condizionali e di ciclo ed altro ancora.

```
ALTER   PROCEDURE dbo.listUsers
AS
SELECT  nome,  email
FROM            Utenti
WHERE
DATEDIFF( d, GETDATE(), dataScadenza ) = 7
```

La sintassi per la cancellazione della stored procedure è invece:

```
DROP PROCEDURE dbo.listUsers
```

E SE CI FOSSERO DEI PARAMETRI?

Una stored procedure accetta parametri in ingresso e può fornire dei parametri di output. Questi sono molto utili per rendere le procedure dinamiche. La sintassi di dichiarazione è molto semplice: supponiamo di voler parametrizzare il numero di giorni per il controllo della data di scadenza:

```
CREATE  PROCEDURE          dbo.listUsersParams
@NumGiorni INT
AS
SELECT  nome, email
FROM    Utenti
WHERE
DATEDIFF( d, GETDATE(), dataScadenza ) =
                                  @NumGiorni
```

A questo punto eseguendo

```
EXECUTE dbo.listUsersParams 7
```

otteniamo la lista degli utenti la cui data di scadenza per una qualche operazione è tra 7 giorni.
Per impostare dei parametri di ritorno occorre utilizzare la parola chiave *OUTPUT.*

```
CREATE  PROCEDURE          dbo.getEmailFromId
                    @idUtente INT,
                    @email VARCHAR(50) OUTPUT
AS
SELECT  @email = email
FROM    Utenti
WHERE   idUtente = @idUtente
```

Abbiamo ottenuto una procedura che ci restituisce l'email di un utente a partire dall'id. Per eseguire la procedura scriviamo:

```
DECLARE     @email VARCHAR(50)
EXECUTE     dbo.getEmailFromId 2, @email OUTPUT
SELECT      @email
```

UTILIZZARE LE STORED PROCEDURE PER AGGIORNARE UNA TABELLA

Possiamo utilizzare delle stored procedure parametriche per inserire o modificare dei record di una tabella. Vediamo subito un esempio di inserimento:

```
CREATE PROCEDURE dbo.InsertUser
            @nome VARCHAR(50),
            @cognome VARCHAR(50),
            @email VARCHAR(50)
AS
INSERT INTO Utenti
(
  [nome],
  [cognome],
  [email]
)
VALUES
(
 @nome, @cognome, @email
)
```

Abbiamo creato una procedura che accetta come parametri *nome, cognome* ed *email.*
La procedura esegue una query di *Insert* sul database creando di fatto un nuovo record.
Per l'esecuzione ci basta scrivere

```
dbo.InsertUser 'Archimede', 'Pitagorico',
                            'achimede@paperopoli.dys'
```

Se vogliamo conoscere anche l'id del nuovo utente inserito occorre scrivere

```
dbo.InsertUser 'Archimede', 'Pitagorico',
                            'achimede@paperopoli.dys'
SELECT @@identity
```

Analogamente possiamo creare una stored procedure per la modifica dei dati di una tabella

```
CREATE PROCEDURE dbo.UpdateUser
            @idUtente INT,
            @nome VARCHAR(50),
            @cognome VARCHAR(50),
            @email VARCHAR(50)
AS
UPDATE Utenti
SET
  [nome]    = @nome,
  [cognome] = @cognome,
  [email]   = @email
WHERE idUtente = @idUtente
```

Per modificare l'utente con *idUtente = 11* ad

SQL MAIL

SQL Mail è uno strumento di SQL Server che consente di inviare e ricevere email. Per funzionare ha bisogno di utilizzare un account di posta che può essere configurato dal pannello di controllo di Windows. Una volta configurato il servizio è possibile utilizzare delle stored procedure quali

xp_startmail
xp_stopmail
xp_sendmail
xp_readmail
xp_deletemail
xp_findnextmsg
sp_processmail

per l'invio e la ricezione di messaggi email.

esempio possiamo scrivere la seguente query

```
dbo.UpdateUser 11, 'Pietro', 'Gambadilegno',
                          'pietro@topolinia.dys'
```

L'inserimento e modifica dei dati tramite stored procedure, da una serie di vantaggi tra cui la possibilità di utilizzare il codice scritto in più progetti e soprattutto l'isolamento dei dati aumentando di fatto la protezione dei dati.

ESEGUIRE STORED PROCEDURE IN .NET

Per eseguire una stored procedure in ambiente .Net possiamo utilizzare le funzionalità fornite dai namespace *System.Data* e *System.Data.SqlClient*. Il namespace *System.Data.SqlClient* ha un oggetto chiamato *SqlCommand*, che consente di eseguire delle query su database. Una volta impostata la connessione, occorre impostare il valore della poprietà *CommandType* dell'oggetto *SqlCommand* a *"StoredProcedure"*.
Vediamo come:

C#

```
SqlCommand cmd  = new SqlCommand(
                          "listUsers", conn);
cmd.CommandType = CommandType.StoredProcedure;
```

VB .Net

```
Dim cmd As SqlCommand = New
                  SqlCommand("listUsers ", conn)
cmd.CommandType = CommandType.StoredProcedure
```

Quando creiamo l'oggetto di tipo *SqlCommad*, dobbiamo passargli come parametro il nome della stored procedure da eseguire.
Impostiamo il *command-type* dopodiché siamo pronti ad eseguire la stored procedure.

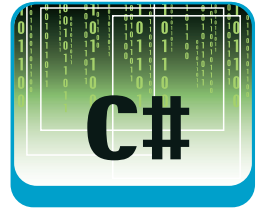
PASSARE I PARAMETRI ALLA STORED PROCEDURE

Per passare dei parametri ad una stored procedure *parametri* occorre utilizzare il metodo *Add* della collection *Parameters*. Il metodo accetta come parametro un oggetto di tipo *SqlParameter* contenente la coppia *"nome parametro"*-*"valore"* come nell'esempio

C#

```
SqlCommand cmd  = new
           SqlCommand("checkInScadenza ", conn);
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.Add(new
           SqlParameter("@nomGiorni", numGiorni));
```

VB.Net

```
Dim cmd As SqlCommand = New
                  SqlCommand("listUsers ", conn)
cmd.CommandType = CommandType.StoredProcedure
cmd.Parameters.Add(new
           SqlParameter("@nomGiorni", numGiorni));
```

UN ESEMPIO COMPLETO

In questo esempio realizziamo un'applicazione console che esegue una stored procedure, ne legge il recordset risultante e stampa i dati a video.

I CURSORI

I Cursori consentono di elaborare un set di risultati di query SQL. In particolare, permettono il posizionamento sulle singole righe del set di risultati e offrono la possibilità di recuperarne il contenuto.
Un cursore viene dichiarato tramite la seguente sintassi:

```
DECLARE NomeCursore CURSOR
FOR QueryDiSelezione
```

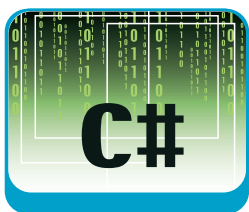
Invece, per il recupero delle tuple, viene utilizzata l'istruzione FETCH che consente di memorizzare i valori della tupla corrente all'interno di variabili.

Vediamo un semplice esempio in cui dichiariamo un cursore che contiene il risultato di una query e lo scorriamo stampandone il contenuto:

```
DECLARE @nome     VARCHAR(50)
DECLARE usersCur   CURSOR
FOR SELECT nome FROM Utenti
OPEN usersCur
FETCH usersCur INTO @nome
WHILE @@FETCH_STATUS  = 0
BEGIN
   PRINT @nome
   FETCH usersCur INTO @nome
END
CLOSE        usersCur
DEALLOCATE usersCur
```

Apriamo Visual Studio e creiamo una nuova console application. Occorre importare i namespace *System.Data* e *System.Data.SqlClient*. La stored procedure che utilizziamo è quella già vista negli esempi precedenti *"checkInScadenza"*. Le operazioni che effettuiamo sono: apertura della connessione, impostazione della stored procedure, esecuzione e visualizzazione dei dati. Alla fine facciamo pulizia degli oggetti aperti.

C#

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace TestStoredProcedure
{
  class TestStoredProcedure
```

```
{
  [STAThread]
  static void Main(string[] args)
  {
    int numGiorni = 7;
    string connStr = "Server=(local);
             DataBase=test;Integrated Security=SSPI";
    SqlConnection dbConn = null;
    SqlDataReader aReader  = null;
    Console.WriteLine("Utenti in scadenza");
    try
    {
      // creo l'oggetto connection
      dbConn = new SqlConnection(connStr);
      dbConn.Open();
      // creo l'oggetto command
      SqlCommand cmd  = new SqlCommand(
                  "checkInScadenza", dbConn);
      // imposto il tipo di comando
      cmd.CommandType =
                 CommandType.StoredProcedure;
      // Aggiungo i parametri
      cmd.Parameters.Add(new SqlParameter(
                  "@NumGiorni", numGiorni));
      // eseguo
      aReader = cmd.ExecuteReader();
      // stampo i risultati
      while (aReader.Read())
      {
        Console.WriteLine(
          "Nome: {0}; Email: {1}",
          aReader["nome"],
          aReader["email"]);
      }
    }
    finally
    {
      if (dbConn != null)
        dbConn.Close();
      if (aReader != null)
        aReader.Close();
    }
  }
}
```

VB.Net

```
Imports System.Data
Imports System.Data.SqlClient

Module TestStoredProcedure

  Sub Main()
    Dim numGiorni As Integer = 7
    Dim connStr As String = "Server=(loca);
            DataBase=test;Integrated Security=SSPI"
    Dim dbConn As SqlConnection = Nothing
    Dim aReader As SqlDataReader = Nothing
    Console.WriteLine("Utenti in scadenza")
    Try
      ' creo l'oggetto connection
      dbConn = New SqlConnection(connStr)
      dbConn.Open()
      ' creo l'oggetto command
      Dim cmd As SqlCommand
      cmd = New SqlCommand("checkInScadenza",
                                       dbConn)
      ' imposto il tipo di comando
      cmd.CommandType =
                  CommandType.StoredProcedure
      ' Aggiungo i parametri
      cmd.Parameters.Add(New SqlParameter(
                    "@NumGiorni", numGiorni))
      ' eseguo
      aReader = cmd.ExecuteReader()
      ' stampo i risultati
      Do While aReader.Read()
        Console.WriteLine("Nome: {0}; Email: {1}",
               aReader("nome"), aReader("email"))
      Loop
    Finally
      If Not dbConn Is Nothing Then
        dbConn.Close()
      End If
      If Not dbConn Is Nothing Then
        aReader.Close()
      End If
    End Try
  End Sub

End Module
```

CONCLUSIONI

Le Stored Procedure rappresentano una soluzione molto flessibile e potente per gli amministratori di database come per i programmatori. Se usate in congiunzione ai JOBS di cui parliamo in questo stesso numero di ioProgrammo consentono un'amministrazione pressoché automatica di ogni operazione legata a SQL Server, dal backup alla manutenzione delle tabelle. La disponibilità di una versione express di SQL Server rappresenta un'enorme ricchezza a cui tutti i programmatori possono attingere per testare i loro programmi. In caso di software che non devono gestire basi di dati sopra i 4 GB, SQL server express rappresenta anche una soluzione operativa. Certo, per la gestione dei JOBS, le utility di reportistica e le altre caratteristiche avanzate, SQL Server Standard edition rappresenta l'unica soluzione possibile in applicazioni di carattere enterprise.

Carmelo Scuderi

SUL WEB

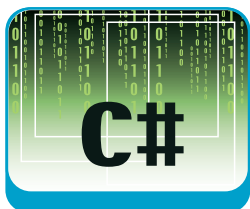
SQL Server Central
http://www.sqlservercentral.com

SqlJunkies
http://www.sqljunkies.com

L'AUTORE

Carmelo Scuderi **è ingegnere informatico. Si occupa di sviluppo software web-based per una società di telecomunicazioni di Milano.**
Gestisce un sito web ricco di script e manuali per chi si affaccia al mondo della programmazione web (www.morpheusweb.it).

FACCIAMO FARE IL "LAVORO" A SQL SERVER

ALLA SCOPERTA DI UNA DELLE FUNZIONALITÀ AVANZATE DEL NUOVO PRODOTTO DI MICROSOFT. SCOPRIREMO COME CON POCO SFORZO SIA POSSIBILE FAR COMPIERE DIRETTAMENTE AL DB OPERAZIONI RIPETIVE

Una delle funzionalità presenti in SQL Server 2005 che maggiormente saranno apprezzate dai programmatori come dagli amministratori è quella relativa ai JOBS. I Jobs sono assimilabili ad una serie di "Macro" che possono essere eseguite direttamente da SQL Server secondo scadenze temporali impostate dall'amministratore di sistema. Utilizzeremo questo articolo per inviare un'email di notifica agli utenti presenti nel database effettuando un controllo su una data di scadenza contenuta in una tabella utenti. Il nostro job sarà eseguito più volte al giorno, e se verificherà che ci sono utenti che hanno una data di scadenza inferiore a una prefissata provvederà a notificargli la scadenza con un'email. In tutto questo non toccheremo una sola riga di codice in nessun linguaggio di alto livello, ma faremo tutto direttamente da SQL.

INIZIAMO

L'elemento principale è l'Agente SQL Server accessibile sotto la cartella Gestione dell'istanza del server.

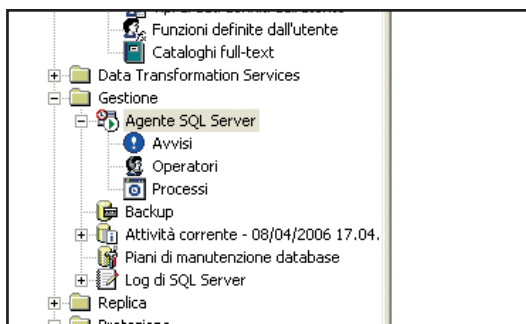



Fig. 1: ***Agente SQL Server***

I Job vengono gestiti tramite una serie di azioni: la definizione del *job*, la gestione delle azioni da eseguire, la pianificazione delle esecuzioni e la definizione dei messaggi di notifica. Per creare un nuovo Job, occorre cliccare con il tasto destro del mouse sulla voce *Job* (o *Processi*) e selezionare *"Nuovo Processo..."* Nel tab *"Generale"* vanno inserite alcune informazioni generali sul *Job* quali: *nome, categoria, proprietario* e *descrizione.*

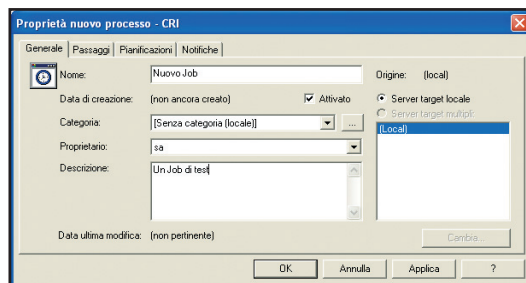



Fig. 2: ***Creazione di un Job***

Una volta inserite le informazioni base è possibile definire una o più azioni da effettuare all'esecuzione del *Job.* Da tab *"Passaggi"* cliccare su *"Nuovo".* Viene visualizzata una maschera in cui è possibile scegliere: un nome per il passaggio, il tipo di passaggio (se uno script T-SQL, un ActiveX, un comando di sistema...), il database su cui lavorare ed infine il comando.

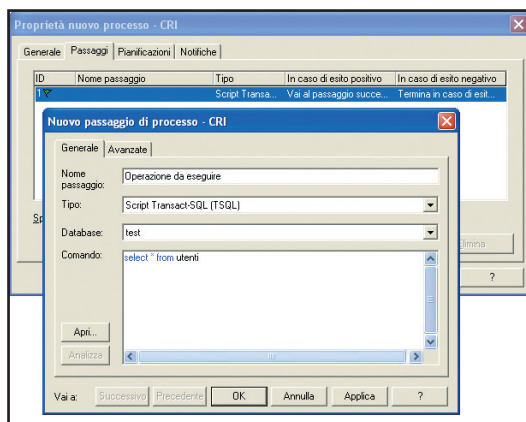



Fig. 3: ***Definizione dei passaggi***

Dal tab *"Avanzate"* è inoltre possibile definire le azioni da eseguire in caso di esito positivo o negativo del passaggio e delle opzioni di eventuali comandi *T-SQL.* La terza operazione da effettuare è la schedulazione del *Job.* Dal tab *"Pianificazioni",* cliccare su *"Nuova pianificazione"* e scegliere il tipo di schedulazione. La maschera è molto intuitiva e non necessita di particolari conoscenze.

Per le pianificazioni periodiche è possibile impostarne la frequenza cliccando sul pulsante *"Cam-*

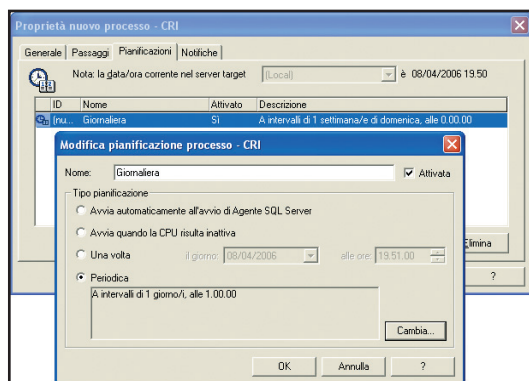



Fig. 4: ***Pianificazione delle esecuzioni***

bia...”. L’ultima azione da eseguire è l’impostazione delle notifiche. Dal tab *“Notifiche”* è possibile scegliere chi notificare ed in che modo farlo in caso in esito positivo o negativo, o semplicemente al termine dell’esecuzione. Creato il *Job*, questo verrà visualizzato nella lista dei *Processi* e potrà essere modificato o cancellato dall’amministratore del database.

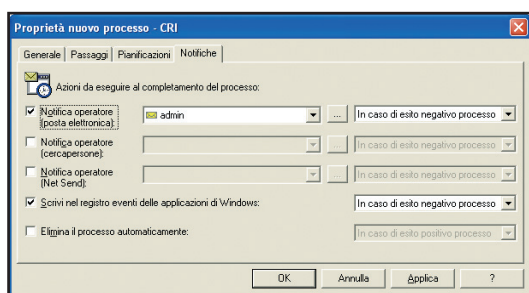



Fig. 5: ***Gestione delle notifiche***

LA NOTIFICA

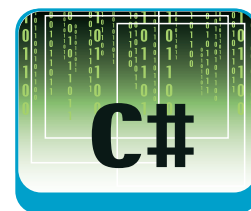
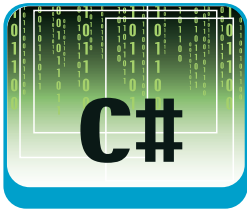
L’esempio ha come prerequisito l’esistenza di una tabella utenti

```
CREATE TABLE [Utenti] (
        [idUtente] [int] IDENTITY (1, 1) NOT NULL ,
        [nome] [varchar] (50) COLLATE
                Latin1_General_CI_AS NOT NULL ,
        [cognome] [varchar] (50) COLLATE
                Latin1_General_CI_AS NOT NULL ,
        [email] [varchar] (50) COLLATE
                Latin1_General_CI_AS NOT NULL ,
        [dataScadenza] [datetime] NOT NULL
            CONSTRAINT [DF_Utenti_dataIscrizione]
          DEFAULT (DATEADD(yyyy, 1, GETDATE())),
        CONSTRAINT [PK_Utenti] PRIMARY KEY
                                        CLUSTERED
        ([idUtente])  ON [PRIMARY]
) ON [PRIMARY]
GO
```

Il nostro *job* invierà un’email di notifica 15 giorni prima della scadenza prefissata nel campo *dataScadenza* ed una 7 giorni prima. L’intero processo verrà gestito tramite una stored procedure parametrica di SQL Server e la creazione di un *Job* che verrà schedulato con frequenza giornaliera e si occuperà dell’esecuzione della stored procedure. Consideriamo una query di base:

```
SELECT  nome, email
FROM     Utenti
WHERE
 DATEDIFF(d, GETDATE(), dataScadenza) = @NumGiorni
```

Questa non fa altro che selezionare gli utenti che hanno come data di scadenza *“Oggi + NumGiorni”.* A questo punto possiamo scrivere la procedura:

```
CREATE  PROCEDURE           [checkInScadenza]
   @numGiorni INT -- La variabile
AS

DECLARE @nome   VARCHAR(50)
DECLARE @email  VARCHAR(50)
DECLARE @msg    VARCHAR(8000)

-- Creiamo un cursore per scorrere il risultato della query
DECLARE          utentiScadenza CURSOR FOR
SELECT  nome, email
FROM     Utenti
WHERE
 DATEDIFF(d, GETDATE(), dataScadenza) = @NumGiorni

OPEN utentiScadenza

FETCH utentiScadenza INTO @nome, @email

-- Per ogni tupla trovata inviamo una e-mail
WHILE @@FETCH_STATUS   = 0
BEGIN
   SET @msg = ''
   SET @msg = @msg + 'Gentile ' + @nome
                     + ',' + CHAR(10) + CHAR(10)
   SET @msg = @msg + 'La tua iscrizione scade tra '
   SET @msg = @msg + CONVERT(VARCHAR(4),
                                        @numGiorni)
   SET @msg = @msg + CHAR(10) + CHAR(10)
   SET @msg = @msg + 'ti invitiamo a rinnovarla ... '
   SET @msg = @msg + CHAR(10) + CHAR(10)
   SET @msg = @msg + 'Morpheusweb.it' + CHAR(10)
   SET @msg = @msg + 'mailto:
            webmaster@morpheusweb.it' + CHAR(10)
   SET @msg = @msg +
            'http://www.morpheusweb.it' + CHAR(10)

   EXEC MASTER.DBO.XP_STARTMAIL
      @user = 'UtenteMailSQL'
   EXEC MASTER.DBO.XP_SENDMAIL
      @recipients = @email,
      @subject = 'Utente in scadenza',
```

```
    @message = @msg
  EXEC MASTER.DBO.XP_STOPMAIL

  -- passiamo alla tupla successiva
  FETCH utentiScadenza INTO @nome, @email
END

CLOSE      utentiScadenza
DEALLOCATE utentiScadenza
```

Abbiamo dichiarato una stored procedure chiamata *checkInScadenza* che accetta come parametro un intero *NumGiorni*. La procedura utilizza un cursore che permette di scorrere le tuple ottenute dalla query che rappresentano gli utenti in scadenza. Per ogni tupla viene composto un messaggio contenente le informazioni da inviare all'utente; vengono quindi richiamate tre stored procedure di sistema: *XP_STARTMAIL*, *XP_SENDMAIL* e *XP_STOPMAIL* che consentono di inviare la mail all'utente in scadenza. Si passa quindi alla tupla successiva, fino alla fine. Dopo l'invio delle mail, il cursore viene chiuso e deallocato. Salvata la stored procedure, procediamo alla creazione del *Job* che la eseguirà con schedulazione giornaliera.

CREIAMO IL JOB

Prima di tutto iniziamo utilizzano l'enterprise manager per dare il via alla procedura di creazione.

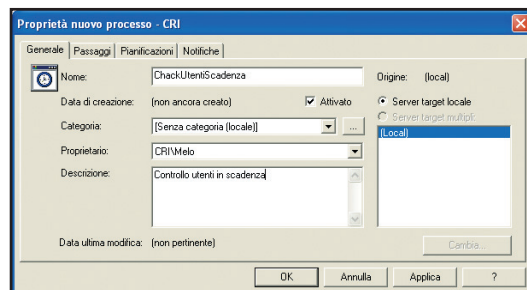



Fig. 6: **Creazione del Job**

Definiamo quindi il passaggio in cui viene richiamata la stored procedure con il parametro impostato la prima volta a 15 e la seconda a 7 in modo da inviare la mail agli utenti in scadenza rispettivamente tra 15 e 7 giorni. Lo script *T-SQL* da inserire è il seguente:

```
DECLARE @RC int
DECLARE @numGiorni int
SET @numGiorni = 15
EXEC @RC = [test].[dbo].[checkInScadenza] @numGiorni
SET @numGiorni = 7
EXEC @RC = [test].[dbo].[checkInScadenza] @numGiorni
```

ricordiamoci di cambiare il database.

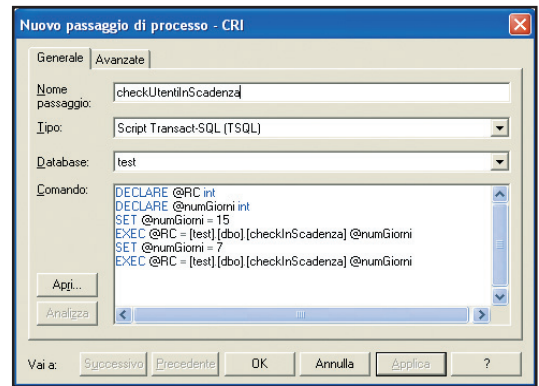



Fig. 7: **Creazione del Job**

Creiamo quindi una nuova pianificazione giornaliera come indicato in **Figura 7**. A questo punto, clicchiamo su *OK* per salvare il *Job*. Da questo momento in poi il nostro *JOB* è impostato, sarà l'apposito servizio ad eseguirlo al momento opportuno

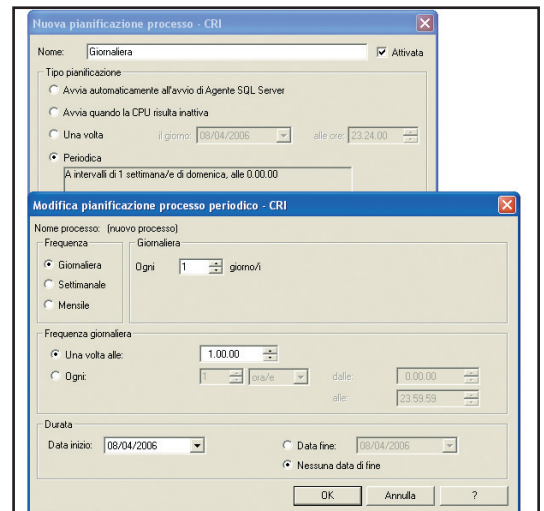



Fig. 8: **Creazione del Job**

AUTOMAZIONE IN SQL SERVER EXPRESS

SQL Server Epress non include l'*Agente* e quindi non è possibile schedulare i *Job* come abbiamo visto. Esiste comunque un workaround che sfrutta lo scheduler di Windows per simularne le funzionalità.

1 Come prima cosa creiamo una query che esegue la stored procedure

```
DECLARE @RC int
DECLARE @numGiorni int
SET @numGiorni = 15
EXEC @RC = [test].[dbo].[checkInScadenza]
                                 @numGiorni
SET @numGiorni = 7
```

```
EXEC @RC = [test].[dbo].[checkInScadenza]
                                     @numGiorni
```

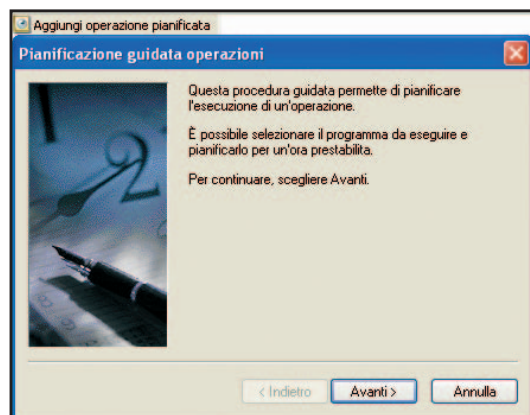
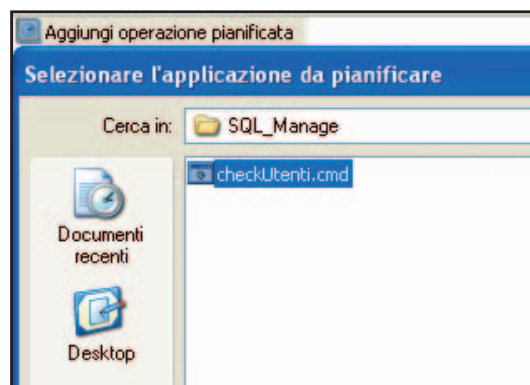
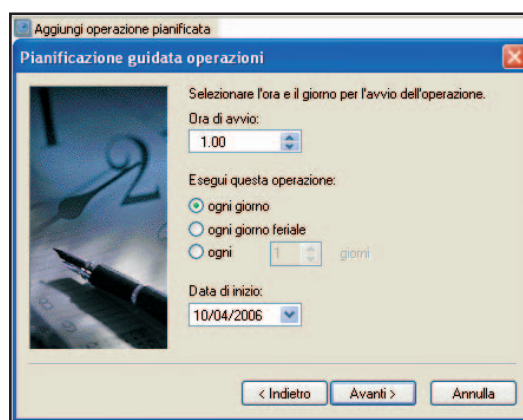
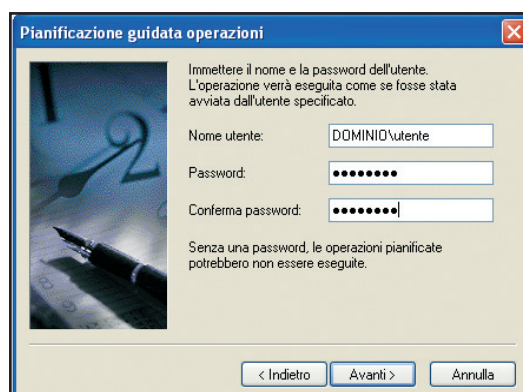
e salviamola ad esempio in *c:\SQL_Manage\Query \checkUsers.sql*. Esiste un eseguibile *sqlcmd.exe* che consente di eseguire dei comandi su SQL Server. La sintassi base per le funzionalità che ci interessano è:

```
sqlcmd [-S server] [–i input file]
```

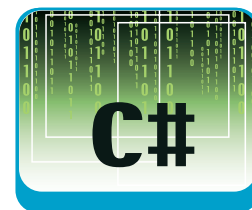
Quindi per eseguire la query appena salvata ci basta digitare:

```
sqlcmd -S.\SQLExpress -i" c:\SQL_ Manage\Query
                                   \checkUsers.sql"
```

Con notepad creiamo un file in cui copiamo l'istruzione vista sopra e salviamolo con nome *c:\SQL_Manage\checkUsers.cmd*.
A questo punto passiamo alla creazione del task di Windows che si occuperà di eseguire il comando secondo le nostre esigenze.

2 Dal pannello di controllo di Windows selezioniamo "Scheduled Task" Facciamo doppio click sull'icona "Add Scheduled Task" e seguiamo la procedura guidata.

3 Come file da eseguire scegliamo *c:\SQL_ Manage\checkUsers.cmd*

4 Diamo un nome alla schedulazione ed impostiamone la periodicità l'orario di esecuzione e la data della prima esecuzione.

8 Impostiamo infine la password e terminiamo la procedura.

CONCLUSIONI

I JOB risultano molto utili in tutte quelle occasioni in cui si vuole automatizzare un compito di sistema. L'assenza di questa funzionalità in SQL Server Express rende la soluzione meno "pulita" tuttavia poter automatizzare i nostri lavori utilizzando questa tecnica renderà senza dubbio più semplice il nostro lavoro.

Carmelo Scuderi

LE CLASSI ED I METODI GENERICI

I GENERICS SONO UNA NUOVA CARATTERISTICA INTERESSANTE DI VISUAL BASIC .NET 2005, PERMETTONO DI RENDERE IL PROPRIO CODICE VELOCE, CONCISO E PIÙ ELEGANTE. SFRUTTIAMOLO A FONDO

Gli insiemi generici, sono così definiti perché al momento della dichiarazione si indica un segnaposto di tipo per gli oggetti contenuti invece di un tipo specifico. Agli oggetti contenuti nell'insieme, viene assegnato un tipo specifico solo quando si crea un'istanza della collezione.
Le classi ed i metodi generici sono riutilizzabili, indipendenti dai tipi e molto più efficaci rispetto alle rispettive controparti non generiche.
In pratica, i generics permettono di scrivere un blocco di codice, che si tratti di una classe collezione o di un metodo, che può operare con tipi differenti di argomenti.
La versione 2.0 della libreria di classi di .NET Framework fornisce un nuovo spazio dei nomi, *System.Collections.Generic*, che contiene numerose nuove classi collection generiche che possono essere specializzate per contenere solo valori di un determinato tipo.

CREARE UNA COLLEZIONE DI TIPI INTEGER

Il modo più semplice per vedere come funzionano gli insiemi generici è tramite un esempio. Vediamo come si può definire una collezione che contenga solo valori di tipo integer:

```
Dichiariamo la collezione colDiInteri
        Dim colDiInteri As New List(Of Integer)
```

La nuova parola chiave *Of* specifica che il tipo generico *List* deve essere specializzato per operare con elementi di tipo *Integer*, e solo con questo tipo di elementi.
Aggiungiamo due valori di tipo *Integer* con il solito metodo *Add* delle collezioni

```
        colDiInteri.Add(25)
        colDiInteri.Add(78)
```

cicliamo su tutti gli oggetti presenti nella collezione con la classica struttura *For Each.. Next* e mostriamo a video gli elementi della collezione

```
        For Each i As Integer In colDiInteri
            MessageBox.Show(i)
        Next
```

Per leggere un elemento si usa la normale sintassi senza la necessità di operatori di conversione di tipo

```
        'Lettura di un elemento
        Dim elemento As Integer
        elemento = colDiInteri(0)
        MessageBox.Show(elemento)
```

Con la parola chiave *Of*, abbiamo indicato che il tipo generico *List* deve essere specializzato per operare con elementi di tipo *Integer*, infatti, qualsiasi tentativo di aggiungere elementi di un tipo differente genera un errore di compilazione. Scrivendo il codice seguente:

```
colDiInteri.Add("pippo")
```

si ottiene l'errore: *"Conversion from string "pippo" to type 'Integer' is not valid"*.
Anche se non risulta evidente dal codice appena scritto, la soluzione basata sui generics risolve il problema delle prestazioni, poiché la collection *List(Of Integer)* memorizza i suoi elementi in variabili *Integer* (o più in generale, in variabili del tipo specificato dalla clausola *Of*), perciò non si verifica più nessuna operazione di boxing.
Complichiamo un pochino le cose, e supponiamo di avere una semplice classe Persona con due proprietà pubbliche, (mostrate come variabili pubbliche per brevità di scrittura del codice, ma non dimentichiamoci mai le buone regole di una corretta programmazione ad oggetti)

```
Public Class persona
    Public Nome As String
    Public CodiceFiscale As String

    Public Sub New(ByVal NomePersona As String,
                ByVal CodiceFiscalePersona As String)
        Nome = NomePersona
        CodiceFiscale = CodiceFiscalePersona
    End Sub

End Class
```

È possibile dichiarare un elenco di persone mediante la classe *List* dell'insieme generico (*List* è la classe generica corrispondente a *ArrayList*), scrivendo il codice seguente:

```
Dim Persone As New List(Of persona)
```

Scrivendo questa singola riga di codice abbiamo dichiarato una collezione tipizzata in modo sicuro, che memorizza solo i tipi persona e fornisce completo supporto *IntelliSense* sugli oggetti *persona* ivi contenuti.
Per popolare la collezione di persone possiamo scrivere il codice seguente:

```
Dim Persona1 As New persona("Federica",
                    "AAABBB21B88D086I")
Persone.Add(Persona1)
Dim Persona2 As New persona("Sara",
                    "AAABBB21K88D086I")
Persone.Add(Persona2)
```

Per mostrare a video le persone contenute nella collezione usiamo, come al solito, la struttura *For Each..Next*

```
For Each Pers As persona In Persone
    MessageBox.Show("Nome: " & Pers.Nome &
            ", CF: " & Pers.CodiceFiscale)
Next
```

Vb .Net 2005 mette a disposizione molti tipi generici, oltre all'oggetto *List* appena descritto:

- **Le collezioni generiche Dictionary(Of K,V) e SortedDictionary(Of K,V)** permettono di creare insiemi hash a tipizzazione forte. *Dictionary* è la classe generica corrispondente a *Hashtable.*

- **Collection** è la classe generica corrispondente a *CollectionBase*. *Collection* può essere utilizzata come classe base, ma a differenza di *CollectionBase* non è una classe astratta ed è quindi molto più semplice da utilizzare.

Fig. 1

- **ReadOnlyCollection** è la classe generica corrispondente a *ReadOnlyCollectionBase. ReadOnlyCollection* non è una classe astratta e dispone di un costruttore che semplifica l'esposizione di una classe *List,* esistente come insieme in sola lettura.

- Le collezioni generiche **Stack(Of T)**, **Queue(Of T)** e **LinkedList(Of T)** si comportano come normali elenchi e stack collegati, fatta eccezione per il fatto che è possibile specificare quali tipi di oggetti vi saranno contenuti. Sono utili per creare una versione più robusta ed efficiente delle altre comuni strutture dati.

TIPI GENERIC

Visual Basic 2005 ci permette di creare dei nostri tipi generic. Quando dobbiamo definire una classe *generics*, è necessario una maniera per differenziare un tipo *generic* come *List(Of T)*, che contiene uno o più parametri di tipo, da un tipo *generic* come *List(Of String)* dove il parametro di tipo è stato sostituito (o collegato) ad un tipo specifico.
Nel primo esempio si parla di definizione del tipo *generic,* tipo generic aperto o tipo *generic* non collegato.
Nel secondo esempio si parla di tipo generic collegato.
Scriviamo un piccolo pezzo di codice che ci permette di definire una semplice classe generica.

```
Public Class ClasseGenerica(Of T)
    Public Campo1 As T
End Class
```

Possiamo notare, come il parametro *generic T* possa essere riutilizzato per definire o istanziare altri tipi *generic.*

In tutti gli esempi di codice, per evitare di scrivere ogni volta il nome completo della libreria, assumiamo che le seguenti istruzioni di *Imports* siano utilizzate a livello di file o di progetto:

```
Imports System
            .Collections
Imports System
        .Collections.Generic
```

Quando si crea un'istanza di una classe generica, si indicano i tipi effettivi che andranno a sostituire i parametri di tipo. Il risultato è una classe indipendente dai tipi, che viene personalizzata in base ai tipi specificati.

```
Dim StrGenerica As New ClasseGenerica(
                                    Of String)
StrGenerica.Campo1 = "Pippo"
Dim IntGenerica As New ClasseGenerica(
                                    Of Integer)
IntGenerica.Campo1 = 10
MessageBox.Show(StrGenerica.Campo1)
MessageBox.Show(IntGenerica.Campo1)
```

Un peculiare problema che ci troveremo ad affrontare lavorando con i *generics,* è che non è possibile fare concretamente alcuna ipotesi sul tipo che verrà passato come parametro del tipo *generic*. Ad esempio, il metodo *New* riceve un elemento di tipo *generic T* ma non può invocare alcun metodo di questo elemento, eccetto quelli ereditati da *System.Object*. Allo stesso modo, non è possibile utilizzare alcun operatore, compresi gli operatori matematici e di confronto, l'operatore *Is,* e l'operatore *IsNot,* senza che venga generato un errore di sintassi. Per ovviare a questi inconvenienti, ci vengono in aiuto i vincoli, come vedremo nel proseguo dell'articolo.

I *generics* non sono un concetto del tutto nuovo nel mondo della programmazione. Infatti, i *generics* sono simili ai *template* C++ , pertanto si potrebbe avere già familiarità se si è lavorato in precedenza con questo linguaggio. Tuttavia, i *generics* .NET hanno diverse caratteristiche e vantaggi che i template C + + non hanno, ad esempio i vincoli.

GENERIC MULTIPLI

Una caratteristica, non indifferente, delle classi *generic* è la possibilità di accettare più di un parametro *generic*. Possiamo, ad esempio, definire una semplice classe che permette di mettere in relazione tra di loro, due oggetti di un determinato tipo:

```
Public Class MembroDi(Of TI, T2)
    Public ReadOnly Oggetto1 As TI
    Public ReadOnly Oggetto2 As T2

    Public Sub New(ByVal objl As TI, ByVal obj2 As T2)
        Oggetto1 = objl
        Oggetto2 = obj2
    End Sub
End Class
```

A questo punto utilizziamo la classe *persona,* definita in precedenza, e definiamo una nuova classe *azienda* che dovrà contenere alcune informazioni tipiche di una azienda:

```
Public Class azienda
    Public RagioneSociale As String
    Public PartitaIva As String

    Public Sub New(ByVal RagSoc As String, ByVal
                                        PIva As String)
        RagioneSociale = RagSoc
        PartitaIva = PIva
    End Sub
End Class
```

La classe *MembroDi* permette di indicare per quale azienda lavora una determinata persona:

```
Dim az1 As New azienda(
                        "GNSoft", "000000000")
Dim pers1 As New persona(
                "Annamaria", "LLLBBB21K88D086I")
Dim Dipendente1 As New MembroDi(
                Of azienda, persona)(az1, pers1)
Dim pers2 As New persona(
                "Patrizia", "PPPBBB21K88D086I")
Dim Dipendente2 As New MembroDi(
                Of azienda, persona)(az1, pers2)
```

In una situazione reale, ci possiamo trovare di fronte alla necessità di gestire molte persone e molte aziende. Per questo motivo possiamo creare una collezione a tipizzazione forte che possa contenere oggetti di tipo *MembroDi*.
Per ottenere questo risultato, si possono utilizzare più *keyword Of* nidificate, aumentando notevolmente la potenza dei *generics:*

```
Dim Dipendenti As New List(Of MembroDi(
                        Of azienda, persona))
Dipendenti.Add(Dipendente1)
Dipendenti.Add(Dipendente2)
```

OVERLOADING ED EREDITARIETÀ

In VB.Net 2005 è possibile definire tipi generic con lo stesso nome ma con un numero differente di parametri *generic*. Ad esempio, le tre classi scritte di seguito possono coesistere nello stesso namespace:

```
Public Class ClasseOverload
    '....
End Class

Public Class ClasseOverload(Of T)
    '...
End Class

Public Class ClasseOverload(Of T, K)
    '...
```

```
End Class
```

Questa caratteristica è simile all'overloading di un metodo, infatti, in fase di compilazione, VB utilizza la classe il cui numero di parametri *generic* corrisponde al numero di argomenti *generic* passati come parametro:

```
Dim c1 As ClasseOverload     ' Una istanza della prima classe.
Dim c2 As ClasseOverload(Of Long)     ' Una istanza della seconda classe.
Dim c3 As ClasseOverload(Of Long, Decimal)     ' Una istanza della terza classe.
```

Da quello detto finora, sembrerebbe che un parametro *generic* possa essere utilizzato in qualsiasi punto di una classe, come se fosse un ordinario nome di tipo, ma questo non corrisponde al vero poiché esistono alcune eccezioni:

- Non è possibile utilizzare un parametro *generic* nella clausola *Inherits*. In pratica, non si può derivare da un tipo passato come parametro *generic*.
- Non è possibile utilizzare un parametro *generic* in una dichiarazione di attributo.
- Non è possibile utilizzare un parametro *generic* per referenziare un'interfaccia nella clausola *Implements*.
- È possibile derivare tipi generici dalla maggior parte delle classi base, nonché utilizzare vincoli per imporre la derivazione di parametri di tipi generici da classi base, ma in questa versione di .NET Framework non sono supportati i tipi generici associati al contesto.
- Non è possibile avere parametri di tipi generici nelle enumerazioni.
- I metodi dinamici *lightweight* non possono essere generici.
- È possibile creare un'istanza di un tipo nidificato in un tipo generico, solo se ai tipi sono stati assegnati i parametri di tipo relativi a tutti i tipi di inclusione.

Il primo limite, vieta quindi di ereditare un tipo da un altro tipo definito per mezzo di un parametro *generic,* però questo non ci vieta, tuttavia, di utilizzare un tipo *generic* nella clausola *Inherits,* come vuole la prassi comune. Ad esempio, si possono definire le seguenti due classi che sono basate sul tipo *MembroDi* definito in precedenza:

```
Public Class RelazioneAziendaPersona
    Inherits MembroDi(Of azienda, persona)
    Public Sub New(ByVal az As azienda, ByVal pers As persona)
        MyBase.New(az, pers)
    End Sub
End Class
Public Class ColRelazioneAziendaPersona
    Inherits List(Of RelazioneAziendaPersona)
End Class
```

METODI GENERIC

VB 2005 permette di utilizzare la keyword *Of* persino nella definizione di un metodo. Possiamo, ad esempio, scrivere la seguente procedura:

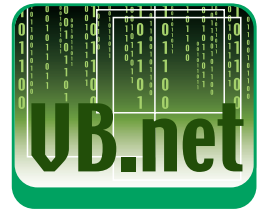
```
Public Sub Procedura(Of T)(ByVal Op1 As T, ByVal Op2 As T)
        '..............
        '..............
End Sub
```

Si può invocare il metodo *Procedura* passando due variabili dello stesso tipo:

```
Dim op1 As Integer = 222
Dim op2 As Integer = 444
Procedura(Of Integer)(op1, op2)
```

Nella maggior parte dei casi, si può invocare un metodo generic senza utilizzare la keyword *Of,* infatti il seguente codice funziona correttamente.

```
Procedura(op1, op2)
```

In alcuni casi è però necessario specificare la clausola *Of,* ad esempio quando i due argomenti sono di tipo differente. In questi casi possiamo evitare l'errore di compilazione in due modi: possiamo convertire manualmente il secondo argomento allo stesso tipo del primo argomento, oppure possiamo specificare la clausola *Of* nell'invocazione del metodo.

In tutte le applicazioni destinate alla versione 2.0 è consigliabile utilizzare le nuove classi di insiemi generiche anziché le controparti non generiche meno recenti, quale *ArrayList*

INSIEMI GENERICI

Gli insiemi generici consentono di ottenere notevoli risparmi di tempo. Chiunque abbia creato insiemi personalizzati conosce la quantità di codice che può essere necessaria. Gli insiemi generici in Visual Basic permettono di creare in una riga di codice l'equivalente di un insieme personalizzato. Non è più necessario creare insiemi personalizzati separati per ciascun tipo di oggetto da memorizzare. È sufficiente fornire il tipo desiderato nel momento in cui viene creata l'istanza dell'insieme generico.

VINCOLI GENERIC

Supponiamo di voler scrivere un metodo *generic*, che restituisca il valore più basso dei parametri passati come argomenti in un array, la classica funzione che calcola il minimo:

```
Public Function Min(Of T)(ByVal ParamArray
                                    values() As T) As T
    Dim risultato As T = values(0)
    For i As Integer = 1 To UBound(values)
        If values(i) < risultato Then risultato =
                                            values(i)
    Next
    Return risultato
End Function
```

Come si può vedere in figura, l'operatore < (minore di) provoca un errore di compilazione, poiché il compilatore non può fare concretamente alcuna ipotesi sul tipo che verrà passato come parametro del tipo *generic*, e quindi non può essere sicuro che questo metodo verrà utilizzato solo con tipi che supportano questo operatore.

TERMINI E DEFINIZIONI RELATIVI AI GENERICS

Il termine *generics* viene utilizzato per indicare classi, strutture, interfacce e metodi dotati di segnaposto (parametri di tipo) per uno o più tipi contenuti o utilizzati in tali elementi. Una classe di insiemi generici può utilizzare un parametro di tipo come segnaposto per il tipo di oggetti in essa contenuti. I parametri di tipo appaiono come tipi dei campi e come tipi di parametri dei metodi della classe. Un metodo generico può utilizzare il relativo parametro di tipo come tipo del valore restituito o di uno dei relativi parametri formali.

Una **definizione di tipo generico** è una dichiarazione di *classe, struttura* o *interfaccia* che funge da modello, con *segnaposto* per i tipi che può contenere o utilizzare. La classe *Dictionary*, ad esempio, può contenere due tipi: *chiavi* e *valori*.
Dal momento che si tratta solo di un modello, non è possibile creare istanze di una classe, struttura o interfaccia che sia una definizione di tipo generico.

I **parametri di tipo generico**, o *parametri di tipo*, sono i segnaposto presenti in un tipo generico o in una definizione di metodo.

Un **tipo generico costruito**, o *tipo costruito*, è il risultato della specifica di tipi per i parametri di tipo generico di una definizione di tipo generico.

Un **argomento di tipo generico** è qualsiasi tipo utilizzato in sostituzione di un parametro di tipo generico.

Il termine generale "*tipo generico*" include sia tipi costruiti sia definizioni di tipo generico.
I vincoli sono limiti imposti su parametri di tipo generico. è ad esempio possibile limitare un parametro di tipo ai tipi che implementano l'interfaccia generica *IComparer* per garantire la possibilità di ordinare le istanze del tipo. È inoltre possibile vincolare i parametri di tipo a tipi che hanno una determinata classe base, hanno un costruttore predefinito oppure sono tipi di riferimento o tipi di valore.
Gli utenti del tipo generico non possono utilizzare in sostituzione argomenti di tipo che non soddisfano i vincoli.
Una definizione di *metodo generico* è un metodo con due elenchi di parametri, ossia uno di parametri di tipo generico e uno di parametri formali. I parametri di tipo possono apparire come tipo restituito o come tipi dei parametri formali.

Come abbiamo visto in precedenza, questo è un problema tipico che si verifica quando si lavora con i *generics*, che si può aggirare applicando un vincolo sul tipo *T*. Nel nostro caso, ad esempio, possiamo imporre che il metodo debba essere invocato solo con tipi che supportano l'interfaccia *IComparable*. Le interfacce generiche *IComparer* e *IEqualityComparer* consentono di definire un confronto di ordinamento o di uguaglianza e forniscono un sistema per ridefinire tali relazioni nel caso di tipi che implementano l'interfaccia in questione. Possiamo scrivere:

```
Public Function Min(Of T As IComparable)(ByVal
                         ParamArray values() As T) As T

 End Function
```

Poiché abbiamo imposto che *T* debba esporre l'interfaccia *IComparable*, il codice nel metodo può invocare con sicurezza il metodo *CompareTo* per calcolare il valore più basso tra quelli passati come argomento:

```
Public Function Min(Of T As IComparable)(ByVal
                         ParamArray values() As T) As T
    Dim risultato As T = values(0)
    For i As Integer = 1 To UBound(values)
        If risultato.CompareTo(values(i)) > 0 Then
                                 risultato = values(i)
    Next
    Return risultato
End Function
```

Per utilizzare la funzione *Min* si può scrivere il seguente codice dove non è necessario specificare la clausola *Of* nella invocazione del metodo

```
MessageBox.Show(Min(25, 10, 65)) 'Visualizza 10
```

Visual Basic 2005 supporta cinque tipi di vincoli:

- **Vincolo di interfaccia:** l'argomento tipo deve implementare l'interfaccia specificata.
- **Vincolo di ereditarietà:** l'argomento tipo deve derivare dalla classe base specificata.
- **Vincolo di classe:** l'argomento tipo deve essere un tipo *reference*.
- **Vincolo di struttura:** l'argomento tipo deve essere un tipo *value*.
- **Vincolo New:** l'argomento tipo deve esporre un costruttore senza parametri (di default).

Luigi Buono

SOFTWARE SUL CD

Microsoft SQL 2005 Server Express Edition

PER LA PRIMA VOLTA IN VERSIONE GRATUITA, ARRIVA IL DB PER LE AZIENDE

Microsoft SQL server 2005 è la nuova versione del database progettato da Microsoft per rispondere alle esigenze di tipo professionale. Le novità introdotte sono veramente moltissime. Prima fra tutte spicca il nuovo formato file: "MDF". Con questa nuova versione i dati possono essere salvati in un file esterno, con estensione MDF. Questo approccio, molto simile a quello usato da Access con i suoi MDB, rende incredibilmente elevata la portabilità da una macchina all'altra. Ulteriori novità importanti sono state introdotte come ad esempio i Service Broker, che consentono di salvare i dati in una coda del SQL Server locale ed effettuare un update sul server remoto

in modo automatico alla prima disponibilità della connettività, e ancora gli Integration Services grazie ai quali è possibile migrare da qualunque database esterno anche proprietario a SQL Server, adottando un linguaggio di trasformazione interno, appunto Integration Services. La versione Express che qui presentiamo è una versione completa, limitata all'uso di 4GB di dati e un GB di Ram, inoltre non è possibile utilizzare le funzionalità di reportistica, il service broker, e gli IS. Tuttavia rappresenta un'ottima soluzione per lo sviluppo di un gran quantitativo di applicazioni e può tranquillamente essere utilizzata anche in fase di produzione

AJAX FOR .NET

IL FRAMEWORK PER CREARE APPLICAZIONI AJAX IN AMBIENTE MICROSOFT

Ajax è una tecnologia che si sta rapidamente affermando nel campo delle WEB Application. Consente di realizzare pagine web dinamiche in grado di aggiornare solo le parti variabili della pagina senza doverla ricaricare interamente. Ovviamente questo comporta un vantaggio immediato in termini di usabilità delle applicazioni, tale che le web application possono assumere il comportamento tipico in termini di interfaccia, di un'applicazione standalone

Directory: /ajaxfornet

ASPECT C++

PROGRAMMARE AD ASPETTI CON C++

La programmazione ad Aspetti nasce dall'esigenza di riunire insieme molti

"spezzoni" di programmi diversi, un po' come avveniva con la programmazione procedurale ma più legata agli eventi. Consente di definire dei JointPoints a cui viene passato il controllo in relazione ad un evento specifico. Questa libreria consente appunto di adottare questo approccio anche con software scritto in C++

Directory: /aspectcpp

AVICREATOR 1.5

PER CREARE PICCOLE ANIMAZIONI DA INSERIRE NEI PROPRI PROGRAMMI

Un utility di corredo, comoda nel caso in cui si vogliano rendere più eleganti le proprie applicazioni aggiungendovi elementi di feedback multimediali. Per

esempio è possibile con questa utility creare facilmente l'animazione dell classico file che viaggia da una cartella all'altra, o altre animazioni del genere, che certamente non influiscono sulla sostanza dell'applicazione ma la rendono più fruibile e certamente più curata

Directory: Avicreator

AXIS 1.3

IL FRAMEWORK PER LA CREAZIONE DI WEB SERVICES IN JAVA

Ormai la programmazione distribuita avviene quasi esclusivamente per mezzo di Web Services. Chi però ha provato a realizzarne qualcuno in Java, ha scoperto che la tecnica non è così semplice. Con Axis tutto diventa meno complesso, interfacce semplificate per la creazione dei WSDL, per la gestione e il deployment del WS rendono comoda la programmazione in Java anche per questa tipologia di applicazioni. Un must per chi adotta la programmazione distribuita

Directory: /Axis

BAMBOO 1.4.4

IL TOOL PER LA PREVALENZA IN .NET

Di prevalenza ne abbiamo parlato ampiamente negli scorsi numeri di ioProgrammo. Si tratta di una tecnica che implementa un approccio nuovo all'uso dei database. In sostanza i dati non vengono realmente salvati sull'hard disk ma mantenuti in memoria, tutte le operazioni vengono ricordate a mezzo di log. Con l'ampliamento della memoria disponibile questo genere di tecnica sta diventando ormai di uso comune e consente una velocità eccezionale d'accesso e update dei dati rispetto alle tecniche tradizionali, di fatto tutte le operazioni avvengono in Ram con l'aumento di velocità che è facile aspettarsi.

Directory:/ Bamboo

BUGTRACKER.NET 2.2.4

PER TENERE SEMPRE SOTTO CONTROLLO I BUG IN FASE DI PRODUZIONE

Molti di voi avranno avuto di utilizzare qualche volta un “bugzilla”. Si tratta di un'applicazione Web grazie alla quale gli utenti possono segnalare un eventuale difetto di funzionamento riscontrato in un software in fase di utilizzo. Bugtracker.NET nasce con lo

stesso scopo, ma si tratta, ovviamente di un'applicazione interamente funzionante in ambienti Microsoft. Tramite questo tipo di tecnica è facilissimo gestire la cronologia e anche l'importanza dei bug, in modo da aumentare la facilità di creazione degli upgrade.

Directory:/ Bugtracker

COMMANDBAR FOR .NET

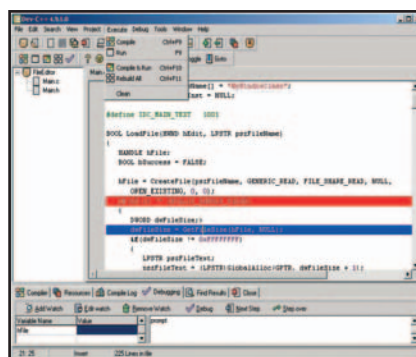
UNA LIBRERIA PER ESTENDERE LE WINDOWS FORM

Un controllo aggiuntivo che consente di implementare all'interno di una Form anche una CoolBar, menu dotadi di Bitmap e context Menu molto particolari. Utile per rendere ancora più interessanti le proprie applicazioni.

Directory:/ CommandBar

DEV C++ 4.9.9.2

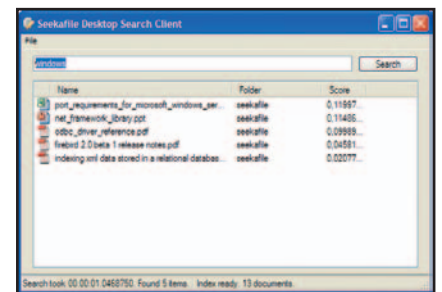
L'IDE PIÙ USATO DAI PROGRAMMATORI C++

Semplice, leggero, completo, Dev Cpp è sicuramente uno degli ambienti più amato da chi sviluppa in C++. Supporta MingGW ma con poche modifiche è possibile adottarlo anche con un compilatore .NET.

Dotato di code complexion e syntax higlighting, sicuramente si pone come un valido aiuto per tutti coloro che sviluppano progetti anche di grandi dimensioni.

Directory: /DevCPP

DOTLUCENE 1.4.3

IL SEGUGIO TROVA TUTTO

Una delle librerie che ha attratto l'attenzione dei programmatori di tutto il mondo nell'ultimo anno è Lucene. Si tratta di una libreria che consente di indicizzare in modo ottimale il contenuto di un Hard Disk per poi poter effettuare delle ricerche molto veloci sull'indice. Lucene è stata utilizzata in

progetti di grandi dimensioni come ad esempio Beagle. Ne esiste anche un porting in ambiente .NET che è appunto quello che vi presentiamo. Utilissima in tutti quei casi in cui si vogliano dotare le proprie applicazioni di funzioni di ricerca “libera” sui contenuti di un supporto o su una parte di esso.

Directory: /DotLucene

DRUPAL 4.7.3 RC3

LA RELEASE CANDIDATE 3 DI UNA DELLE PIATTAFORME PER BLOG PIÙ DIFFUSE

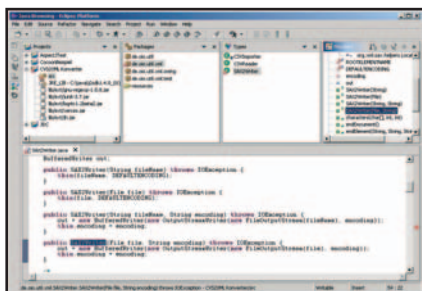
Il fenomeno dei Blog ha cambiato radicalmente la struttura di Internet. Drupal è una delle piattaforme che più di ogni altra si è distinta nella capacità di offrire all'utente un'interfaccia comoda a supporto di funzionalità molto complesse. D'altra parte mette a disposizione del programmatore una serie di API tale che sviluppare un modulo per estenderne le funzionalità diventa divertente oltre che flessibile. Si tratta dunque di una web application che farà ancora parlare di se, e sopratutto garantisce agli

sviluppatori di poterla personalizzare in modo molto efficace per le proprie esigenze

Directory:/ Drupal

ECLIPSE 3.1.2

L'IDE TUTTOFARE

Dalla sua parte un'enorme flessibilità. L'IDE multipiattaforma più utilizzato dai programmatori java, grazie alla sua modularità, sta diventando in breve tempo il punto di riferimento per una larga schiera di sviluppatori.

Attualmente esistono plugin per sviluppare in PHP, per la progettazione di WEB Services, per l'integrazione in Hibernate ed addirittura si stanno studiando soluzioni per Actionscript.- Eclipse si propone come un'ambiente ormai solido e maturo, pronto per essere utilizzato in ogni ambito della programmazione

Directory:/ Eclipse

ECLIPSE ME

LO SVILUPPO MOBILE SECONDO ECLIPSE

EclipseME è un plug-in per Eclipse che vi aiuta nella creazione di una MIDlet. Una Midlet è tipicamente un programma in grado di girare su dispositivi mobili. Eclipse ME aiuta sia nello sviluppo sia nel deployment garantendo una maggiore facilità nella programmazione

Directory:/ EclipseME

FREETTS 1.1.2.1

UN SINTETIZZATORE VOCALE SCRITTO INTERAMENTE IN JAVA

A freeTTS abbiamo dedicato ampio spazio nei numeri precedenti di ioProgrammo. Si tratta di un motore sintesi vocale interamente scritto in Java. Dove per sintesi vocale si intende che questo engine consente di creare applicazioni Java in grado di produrre dei suoni molto simili alla voce umana. Utile ad esempio per farvi leggere le email, oppure un racconto, la quantità di applicazioni disponibili dipende solo dalla vostra fantasia

Directory:/ FreeTTS

GSOAP C++

UNA LIBRERIA PER SCRIVERE WEB SERVICES IN C++

Web Services per .NET per Java, per PHP, ma se ne volessimo scrivere uno in C++? Arriva Gsoap, che appunto ci semplifica la vita in questo compito. Supporta interamente SOAP e la creazione del WSDL. Si tratta di una libreria piuttosto avanzata, anche se ancora in fase embrionale. Non le mancano tuttavia le potenzialità per diventare un prodotto di enorme successo

Directory:/ Gsoap

HIBERNATE 3.2.0

IL TOOL PER LA PERSISTENZA DEI DATI IN JAVA

I programmatori sono abituati a pensare in termini di classi ed oggetti. Non c'è programma che prescinda ormai dalla logica della programmazione OOP. Tuttavia i database SQL ragionano ancora in termini di puro linguaggio, non c'è nessuna correlazione fra un dato e l'oggetto che lo rappresenta. Hibernate aggira questo ostacolo, ponendosi come framework che maschera un database relazionale con una logica OOP.

In questo modo i programmatori Java possono ad esempio accedere a un qualunque database pensando alle righe e alle colonne che lo rappresentano in termini di oggetti e non di entità relazionali. Quella che vi presentiamo in questo numero è la versione più aggiornata del tool, più veloce e affidabile

Directory:/Hibernate

IRRLICHT 1.0

ACCENDI IL MIGLIOR MOTORE 3D OPEN SOURCE!

IrrLicht è un motore per la grafica tridimensionale, scritto in C++ e utilizzabile sia con questo linguaggio, sia con i linguaggi di .NET. Presenta le principali caratteristiche che si trovano anche nei motori professionali e vanta una notevole comunità di sviluppatori, con diversi progetti in attivo. Irrlicht è sicuramente molto veloce, flessibile e in grado di gestire mesh provenienti da diversi ambienti. In ioProgrammo più volte abbiamo proposto articoli completi che mostravano l'utilizzo di questo eccezionale engine per lo sviluppo di videogames

Directory:/ Irrlicht

J2SE 1.5.0 UPDATE 6

L'SDK ESSENZIALE PER LA PROGRAMMAZIONE JAVA

Se siete dei programmatori Java o aspirate a diventarlo non potete mancare di installre il J2SE di Sun, al cui interno è contenuto il compilatore nonché tutte le librerie essenziali per potere programmare in Java.

Quella che vi presentiamo è l'ultima release aggiornata del compilatore che copre molti bachi di programmazione e si presenta leggermente più veloce.

Directory:/ J2SE

JAMELEON 3.2

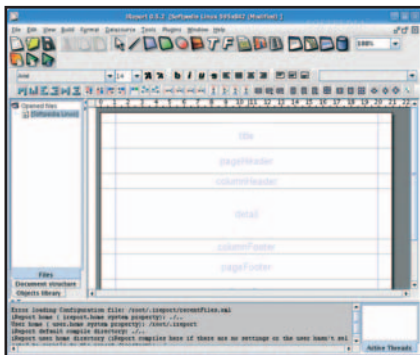
UN FRAMEWORK AUTOMATIZZATO PER IL TESTING DELLE APPLICAZIONI

Jameleon è un tool che consente di testare con efficacia il software da voi creato. Il concetto principale su cui Jameleon si basa è quello de Tag che rappresentano una schermata dell'applicazione. Ciascun gruppo di Tag può essere parametrizzato con dati e chiaramente deve restituire un certo output. Infine tutto può essere automatizzato producendo degli script che salvano i risultati in file di log

Directory:/ jameleon

JASPER REPORTS 1.2.1

CREA I TUOI REPORT A PARTIRE DA APPLICAZIONI JAVA

Forse il miglior tool per la creazione di reportistica da associare ad applicazioni Java. Consente di creare rapporti in formato PDF, HTML, XLS,

CSV e XML. Si tratta di un tool complesso ma anche molto efficace che affianca uno strumento per la creazione dei report a sofisticate API che possono facilmente interfacciarsi con le vostre applicazioni

Directory: /JasperReports

JDEBUG TOOL

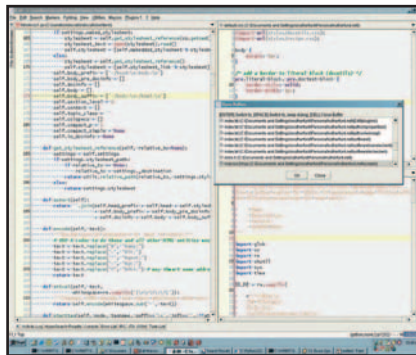
UN DEBUGGER PER JAVA FULL OPTIONAL!

Un debugger standalone costruito sulla base della JPDA, Java Platform Debugger Architecture. L'interfaccia grafica ed un Help ottimamente strutturato consentono un rapido utilizzo del tool. Tra le funzioni segnaliamo: debug remoto, debug multi-thread, modifica delle variabili "al volo", visualizzazione delle classi attualmente caricate, valutazione dei toString, monitoraggio dell'occupazione di memoria, sincronizzazione con gli eventi di load e unload delle classi. In questa versione si apprezza particolarmente la velocità di esecuzione.

Directory:/ Jdebugtools

JEDIT 4.2

L'EDITOR LEGGERO PER PROGRAMMATORI JAVA

Realizzato completamente in Java, jEdit è un completo editor di testi per programmatori disponibile per numerose piattaforme tra cui MacOS X, OS/2, GNU/Linux (Unix) e Windows. Il programma, per l'immediatezza e la semplicità di utilizzo, è molto usato per scopi didattici e da programmatori non professionisti.

jEdit integra un completo linguaggio di macro e dispone di un'architettura facilmente estensibile mediante l'utilizzo di numerosi plugin facilmente gestibili tramite il "plugin manager". Inoltre, nonostante sia stato creato per sviluppare prevalentemente applicazioni Java, dispone di funzionalità per indentare ed evidenziare il codice per oltre ottanta linguaggi di programmazione. Per poter installare e utilizzare jEdit è necessario disporre del J2SDK 1.3 o 1.4.

Directory:/ Jedit

JGAP 2.6

LA LIBRERIA PER GLI ALGORITMI GENETICI

Gli algoritmi genetici rappresentano un campo in continua evoluzione. Si tratta di una metodologia che consente di sviluppare soluzioni evolutive sulla base di combinazioni cromosomiche. La realtà è meno complessa delle parole utilizzate per descriverla, sostanzialmente si parte da una popolazione di soluzioni che vengono rimescolate simulando l'evoluzione del patrimonio genetico, dal nuovo nucleo di soluzioni si estrapolano quelle migliorative delle precedenti e si continua così fino alla completa risoluzione del problema. jGap è una libreria Java che mette a disposizione diverse primitive proprio per la gestione degli algoritmi genetici, che ormai trovano applicazione in più di un settore.

Directory:/ Jgap

JOOMLA 1.0.8

L'EREDE DI MAMBO

Molti di voi conosceranno Mambo, si tratta di uno dei CMS più utilizzati in rete. A costruire il successo di Mambo è stata la sua alta modularizzazione, la capacità di consentire agli

utenti di personalizzare il sistema con poche, rapide operazioni, la completezza del sistema. Peccato che recentemente gli sviluppatori di Mambo si siano trovati n disaccordo con le scelte di Mirò la software house che ha prodotto il software fino ad ora. Così molti di loro hanno abbandonato Mirò e hanno dato vita a Joomla l'erede di Mambo. Basato sul suo stesso codice ma con la promessa di essere OpenSource ora come nelle future versioni.

Directory:/ Joomla108

JUNIT 3.8.2

PER TESTARE FACILMENTE PROGRAMMI JAVA

Uno dei tool fondamentali per procedere al testing di programmi privi di Bug. Junit consente di creare test automatizzati che possono mettere sotto stress le vostre applicazioni e creare report affidabili che vi consentono di intervenire in modo efficace su ogni eventuale bug

Directory:/ Junit

ALBERI DI RICERCA

LA STRUTTURA DATI ALBERO È UNA DELLE PIÙ IMPORTANTI NEL PANORAMA DELLA PROGRAMMAZIONE. IN QUESTO ARTICOLO IMPAREREMO COME GESTIRE AL MEGLIO LE TECNICHE CHE NE REGOLAMENTANO IL FUNZIONAMENTO

Una delle ragioni per le quali si applica un criterio di ordinamento ad una serie di dati è quella di voler successivamente eseguire su questa serie delle ricerche rapide. Con insiemi di dati disordinati si può procedere alla ricerca soltanto in modo del tutto sequenziale, ovvero esaminando i dati uno dopo l'altro, a partire dal primo, fin quando non si trova quello cercato. Viceversa nel caso in cui una serie di dati sia stata in qualche modo ordinata si può applicare una ricerca binaria. Con tale metodo i tempi di computazione si abbattono. La logica di esplorazione binaria delle informazioni ha introdotto la necessità di costruire delle strutture dati a cui questo tipo di attività si adatti facilmente. Sono così stati introdotti gli alberi, oggetto della presente esposizione, strutture dati solitamente implementate in modo dinamico. I campi di applicazione sono tanti, si pensi al backtracking e a molti algoritmi ricorsivi; la soluzione in tale ambito non è altro che un nodo da ricercare all'interno di un albero. Molti metodi della ricerca operativa fanno riferimento ad alberi, a titolo esplicativo vale branch and bound. E, senza andare lontano, si faccia riferimento al *radix sort* oggetto delle nostre attenzioni nello scorso appuntamento. E così si potrebbe continuare a lungo. Ad ogni modo nel corso dell'articolo faremo alcuni esempi.

RICERCA BINARIA

Assumiamo che un insieme di n dati sia strutturato come array ordinato. È possibile applicare una ricerca binaria con il risultato di ottenere al più *lg(n)* confronti, con *lg* logaritmo in base 2. Nel caso della ricerca sequenziale, ossia il confronto in sequenza dal primo elemento fino a quello che si intende trovare, si devono fare nel caso peggiore n confronti. Il metodo è semplice. Si divide l'array a metà e si confronta l'elemento *x* che si vuole cercare con l'elemento centrale dell'array. Se sono uguali siamo stati molto fortunati, e la ricerca è conclusa, altrimenti si procede con l'analisi del confronto dei due elementi. Se *x* è minore dell'elemento corrente dell'array vuol dire che il valore cercato si trova nella metà inferiore dell'array, altrimenti ancora, l'elemento sarà collocato nella metà superiore dell'array. Con un solo confronto abbiamo quindi eliminato una metà di array. Il metodo prosegue applicando lo stesso procedimento alla parte di array rimasta. Rapidamente, al più in *lg(n)* passaggi, la ricerca termina. Per l'implementazione sarà sufficiente mantenere tre indici. I primi due che chiameremo inizio e fine circoscrivono il vettore parziale che iterazione dopo iterazione si dimezza; il terzo: centro, indica l'elemento da confrontare con *x*. Ecco il codice C++.

```
// .... Dichiarazione
//  ... input del vettore
//  .... Ordinamento del vettore
cout<<"\n x : ";
cin>>x;
inizio=1;
fine=n;
trovato=false;
while ((! trovato) && (inizio<=fine))
{  centro=(inizio+fine)/2;
   if (vettore[centro]==x) trovato=true;
   else if (vettore[centro]>x) fine=centro-1;
        else inizio=centro+1;
};
if (trovato) cout<<"\n il nome è presente in
                              posizione "<<centro;
else cout<<"il nome non è presente";
// ...
```

La logica binaria di tale ricerca ha contribuito alla costruzione di strutture particolari che mettono ancora in risalto le capacità elaborative di questo pur semplice metodo, in particolare: gli alberi.

DEFINIZIONE DI ALBERO

L'albero è una importante e molto usata struttura dati. Usualmente viene implementato con componenti dinamiche costituite da nodi che contengono puntatori. In particolare un albero è costituito da nodi. Ogni nodo contiene una parte informativa e dei collegamenti verso altri nodi. Ogni nodo può quindi avere dei nodi discendenti anche chiamati figli e nodi di riferimento padre di cui è appunto discendente. L'intera struttura è identificata da un nodo origine chiamato radice. L'analogia con il vero albero, appartenente al mondo vegeta-

le per intenderci, è sempre presente. I nodi terminali che non hanno figli sono le foglie. Tutti gli altri sono nodi intermedi o semplicemente nodi. La struttura è organizzata per livelli. Al livello zero troviamo la radice. I discendenti della radice sono a livello uno. I figli dei figli della radice sono a livello due e così via. Man mano che si scende verso livelli maggiori si ha un aumento esponenziale del numero dei nodi.
Ecco una definizione ricorsiva di livello: *"il livello di un nodo in un albero è pari a 1 più il livello del nodo padre, intendendo che la radice ha livello 0"*.
Il numero massimo di nodi per livello si ha quando tutti i nodi hanno il massimo dei figli tale numero è: *m^n*. Il termine m è definito come grado dell'albero, ovvero il numero massimo di nodi discendenti da un generico nodo, *n* è il livello. Il grado di un nodo è il numero di nodi discendenti da esso. Il tipo di albero più usato afferisce al grado due e si tratta proprio dell'albero binario, oggetto dei nostri approfondimenti. Non vanno comunque trascurati altri tipi di alberi che hanno grado maggiore; il conosciuto *B-tree* usato nell'ambito dell'indicizzazione dei file ha grado maggiore di 2. Diamo una definizione formale di albero. Ci servirà in seguito per comprendere alcune funzioni di manipolazione. Un albero è:

1. Una struttura vuota.
2. Un nodo di tipo *A*.
3. Un determinato numero di sottoalberi riferiti ad un nodo di tipo *A*, i sottoalberi sono degli alberi.

Il terzo punto evidenzia la natura ricorsiva dell'albero. Occorre precisare che con il termine *"riferito"* si intende un collegamento da un nodo *padre* verso il nodo *soggetto*.

ALBERO BINARIO

L'albero in cui i nodi hanno il grado due, è detto binario. In tal caso il numero massimo di nodi è 2^n con *n* livelli. Formula ricorrente questa per i programmatori! L'albero binario è il più usato non solo perché è strettamente relazionato alla ricerca binaria ma anche per altre applicazioni. Ad esempio, le espressioni matematiche e logiche trovano una naturale descrizione alberi binari. In **Figura 1** sono mostrate due espressioni logiche rappresentate da altrettanti alberi binari. Il particolare albero binario che sarà oggetto dei nostri approfondimenti è detto di ricerca. Si tratta di un albero in cui le informazioni vengono collocate nei singoli nodi in modo ordinato. Significa che per un qualsiasi nodo il discendente destro avrà rispetto ad esso un contenuto informativo minore, mentre il nodo sinistro

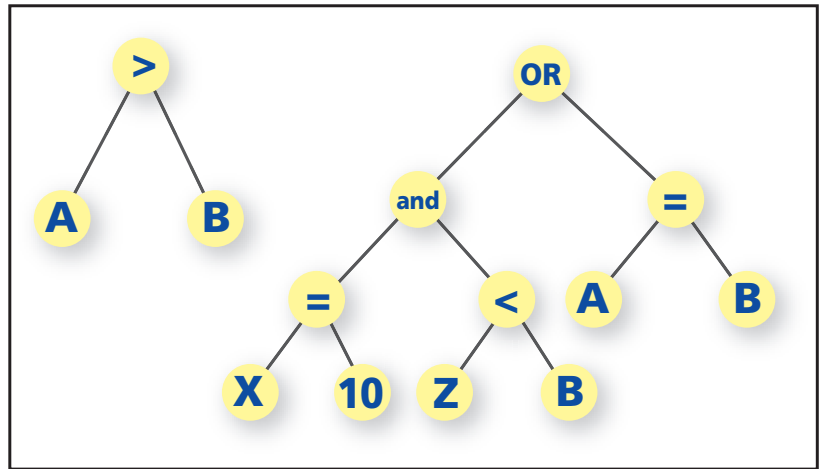



Fig. 1: **Alberi binari che esprimono le due espressioni condizionali: 1- A>B; 2- ((X=10) AND (Z<B)) OR (A=B)**

maggiore. Una struttura così costruita garantisce la ricerca binaria. Nella classe C++ che svilupperemo introdurremo i metodi fondamentali per la manipolazione degli alberi e si porrà il problema di come esplorarli. Esistono tre modi. A seconda dell'ordine di visita. Vedremo tutti e tre i metodi. Il preordine visita in ordine: la radice, il sottoalbero sinistro e il sottoalbero destro. Il inordine visita in sequenza: il sottoalbero sinistro la radice e il sottoalbero destro. Il postordine ha come successione di visite: il sottoalbero sinistro, il sottoalbero destro e la radice. Rispetto all'albero di **Figura 2** i metodi di visita producono i seguenti risultati:

1. **preordine:** *lillo, huggy, mary, molly, navy;*
2. **inordine:** *huggy, lillo, mary, molly, navy;*
3. **postordine:** *huggy, molly, navy, mary, lillo.*

Tutti i metodi hanno le proprie peculiarità, ad esempio il metodo inordine o simmetrico visualizza le informazioni in ordine come il nome stesso ci suggeriva; purché si tratta di un albero di ricerca.

ALBERO COME GRAFO

Un albero è un grafo orientato costituto da nodi collegati a altri nodi discendenti detti figli. Per il discendente il nodo generatore è padre. Tutti i nodi hanno figli tranne le foglie e tutti i nodi hanno padri tranne il nodo generatore radice.

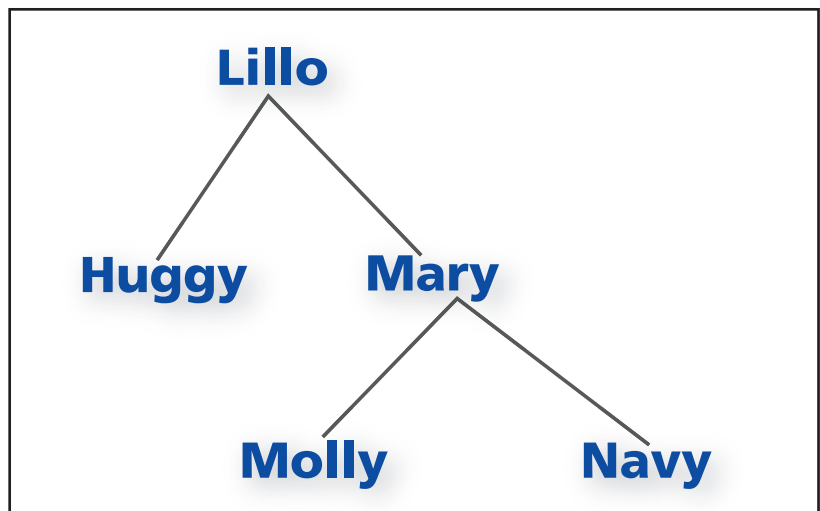



Fig. 2: **Alberi binari e metodi di esplorazione**

Ma passiamo al codice. Definiamo una classe in modo tale che possa essere in futuro facilmente ampliata con i metodi, che sono tanti, per la manipolazione degli alberi. Ecco una idea:

```
// Classe albero
class albero
{
  public: albero();
        albero(char*);
        ~albero();
     void aggiungi(char*);
     void postordine();
     void preordine();
     void inordine();
     bool ricerca(char*);
  private:
  struct nodo
  {
     char *info;
     struct nodo *dx;
     struct nodo *sx;
  };
  struct nodo *tree;
  void agg(nodo* &, char*);
  void postordinepriv(nodo*);
  void preordinepriv(nodo*);
  void inordinepriv(nodo*);
  void cerca(nodo*, char*, bool &);
};
```

Fin qui troviamo la definizione "minima" della classe. Valuteremo anche gli interessanti aspetti forniti dall'applicazione della OOP. Il nome della classe è albero. Rispetto ad essa si trova una parte privata, costituita dalla struttura fisica dell'albero, e da alcune funzioni di base e una parte pubblica che mette a disposizione del programmatore i metodi per la manipolazione dell'albero. La struttura è costituita da un elemento di base che è il nodo fatto di una parte informativa e due puntatori, uno destro *(dx)* e uno sinistro *(sx)* al nodo stesso. Inoltre, è presente il puntatore *tree* al nodo, con esso si identifica la struttura fisica albero. Le tre funzioni private *postordinepriv, preordinepriv* e *inordinepriv* realizzano ricorsivamente le ricerche sull'albero. La loro implementazione è riportata di seguito. Le altre funzioni si occupano di altrettanti aspetti di base come l'aggiunta di un elemento nell'albero e la ricerca. Tra le parti pubbliche vi sono oltre ai costruttori e distruttori le funzioni per accedere alla struttura. Per cui si hanno a disposizione: metodi per esplorare l'albero, che nella loro implementazione fanno riferimento ai corrispondenti metodi privati; metodi per la ricerca e altri per l'inserimento. Di seguito sono riportate le relative implementazioni. Analizziamo dapprima i costruttori e i distruttori.

```
// Implementazione dei costruttori e dei distruttori
albero::albero()
{
    tree=0;
};

albero::albero(char *messaggio )
{
  tree=new nodo;
  tree->info=messaggio;
  tree->dx=0;
  tree->sx=0;
};

albero::~albero()
{
   delete tree;
};
```

I due costruttori generano il nodo radice. Il primo è senza parametri e si limita a puntare la struttura tree a 0 (analogamente si poteva usare l'identificatore *NULL*). Il costruttore con parametro colloca nel primo nodo opportunamente allocato (con *new*) il messaggio (relativo parametro) nella parte informativa del nodo. Il distruttore libera la memoria con delete. Esaminiamo adesso come si possa aggiungere un nodo, in modo che l'albero rimanga sempre ordinato. Tale ordinamento preserva quindi le caratteristiche di albero di ricerca.

```
// privata
void albero::agg(nodo* &nod, char* mess)
{
   if (nod==NULL)
   {
       nodo* p=new nodo;
       p->info=mess;
       p->dx=NULL;
       p->sx=NULL;
       nod=p;
    }
    else if (mess<nod->info) agg(nod->sx,mess);
         else if (mess>nod->info) agg(
```

GLOSSARIO

ALBERO AVL

È un particolare albero binario bilanciato che ha la caratteristica che preso un qualsiasi nodo la differenza di altezza dei due sottoalberi differisce al più di uno. L'altezza è la più grande distanza tra la il nodo e le foglie, in termini di livelli. Per la natura ricorsiva dell'albero si può considerare la radice anziché un nodo qualsiasi proposto nella definizione. Un albero siffatto garantisce ricerca, eliminazione e inserimento con una complessità *O(lg(n))*.

B-TREE

Sono alberi generali con grado qualsiasi che seguono precise caratteristiche.
Sia *A* un *B-Albero* costituito dalle seguenti parti:

- **radice(A), il nodo che identifica la struttura e da cui discendono tutti gli altri nodi.**
- **x, un generico nodo che può contenere da *t-1* a *2t-1* chiavi. Fa eccezione la radice che può avere anche solo una chiave.**
- **foglie, le foglie sono particolari nodi terminali tutte allo stesso livello.**

Vengono usati nella indicizzazione dei file per sfruttare appieno il metodo l'accesso diretto ai file.

```
                                  nod->dx, mess);
};
//pubblica
void albero::aggiungi(char* messaggio)
{
   agg(tree, messaggio);
};
```

Il metodo pubblico semplicemente richiama il corrispondente privato specificando il messaggio da inserire e l'albero (tree che è la struttura privata) a cui fare riferimento. Questo passaggio consente di fare riferimento alla struttura solo nella fase privata. Ciò è necessario per potere applicare la ricorsione. Nel metodo pubblico, come le elementari regole di programmazione ad oggetti (OOP) consigliano, non apparirà la struttura albero su cui si sta facendo l'operazione. Tale metodo è stato usato anche per le altre funzioni ricorsive di visita dell'albero. In *agg* si richiama ricorsivamente la funzione sul sottoalbero destro o sinistro a seconda del risultato della comparazione tra il messaggio e l'elemento informativo chiamato. Nella serie di chiamate ricorsive ad un certo punto ci si imbatterà in una foglia che non ha quindi figli. Qui viene allocato un nuovo nodo e collegato al nodo padre. Con il controllo anche sull'*else* si evitano di duplicare stesse informazioni. In altre parole una stringa non si può presentare nell'albero più di una volta. Da notare la presenza nella funzione agg del simbolo di *reference* & che consente di trattare il parametro associato di tipo variabile, in modo da stabilizzare il suo cambiamento dell'albero.
Esaminiamo le visite.

```
void albero::postordinepriv(nodo *nod)
{
   if (nod!=0)
   {
      postordinepriv(nod->sx);
      postordinepriv(nod->dx);
      cout<<nod->info<<"\n";
   };
};
void albero::postordine()
{  postordinepriv(tree);};
void albero::preordinepriv(nodo *nod)
{  if (nod!=0)
   {  cout<<nod->info<<"\n";
      preordinepriv(nod->sx);
      preordinepriv(nod->dx);
   };
};
void albero::preordine()
{  preordinepriv(tree);
};
void albero::inordinepriv(nodo *nod)
{  if (nod!=0)
```

```
   {  inordinepriv(nod->sx);
      cout<<nod->info<<"\n";
      inordinepriv(nod->dx);
   };
};
void albero::inordine()
{
   inordinepriv(tree);
};
```

Si implementano naturalmente in modo ricorsivo. Si differenziano dal punto in cui viene posta in output la parte informativa. Ad esempio, per la vista preordine si ha prima l'output e poi la chiamata ricorsiva ai due sottoalberi sinistro e destro. Infine, esaminiamo la ricerca in modo da chiudere il cerchio rispetto ad una ricerca fatta su un vettore ordinato nel primo paragrafo.

```
// Ricerca nell'albero
void albero::cerca(nodo* nod, char* mess, bool &tr)
{  if ((nod!=0) && (!tr))
      if (nod->info==mess) tr=true;
      else if (mess<nod->info) cerca(
                                  nod->sx, mess, tr);
         else cerca(nod->dx, mess, tr);
};
bool albero::ricerca(char* messaggio)
{  bool trovato=false;
   cerca(tree, messaggio, trovato);
   return trovato;
}
```

Anche qui si ha una procedura ricorsiva cerca. Si termina la chiamate delle chiamate ricorsive quando si raggiungono tutte le foglie quindi *nod!=0* risulta falso oppure quando trovato (paramentro *tr*) è vero. Nella condizione opposta, si rimane nella procedura. Si controlla poi se la parte informativa è uguale al messaggio, in tal caso si setta a vero la booleana *tr*. Altrimenti si esplora il sottoalbero sinistro o destro a seconda del risultato del confronto.

CONCLUSIONI

Gli alberi sono un fondamentale strumento nelle mani del programmatore. Costruire una classe albero arricchendola dei metodi che possono servire per il particolare uso che ne può scaturire è un utile esercizio. La prossima volta affronteremo l'importante estensione ed evoluzione degli alberi binari, quelli bilanciati. Si tratta di strutture dati più difficili da trattare ma che garantiscono notevoli performance nella manipolazione e ricerca delle informazioni.

Fabio Grimaldi

ALBERO SPLAY

Uno albero *splay* è un albero per ricerca binaria bilanciata con la proprietà, che gli elementi che vengono accessi più frequentemente, si trovano più vicini alla radice, di modo che la loro ricerca duri meno. Per ottenere questa proprietà, si espande (in inglese *splay*) l'albero in modo che ogni chiave cercata, venga spostata alla radice attraverso delle rotazioni. Queste rotazioni oltre a portare il nodo cercato alla radice, accorcia la distanza tra la radice e tutti i nodi che si trovavano tra il nodo cercato e la precedente radice, di circa la metà.